Security of Cryptographic Primitives in Advanced Security Notions

Vom Fachbereich Informatik der
Technischen Universität Darmstadt genehmigte

Dissertation

zur Erlangung des Grades
Doctor rerum naturalium (Dr. rer. nat.)
von

Patrick Struck, M.Sc.
geboren in Bad Soden.

Referierende:	Prof. Dr. Thomas Schneider (Technische Universität Darmstadt) Prof. Dr. Juliane Krämer (Universität Regensburg) Prof. Dr. Gorjan Alagic (University of Maryland)
Tag der Einreichung:	07.02.2022
Tag der mündlichen Prüfung:	30.03.2022

Darmstadt 2022
D 17

ERKLÄRUNG

Hiermit erkläre ich, dass ich die vorliegende Arbeit - abgesehen von den in ihr ausdrücklich genannten Hilfen - selbständig verfasst habe.

Patrick Struck

ACKNOWLEDGEMENTS

No one who achieves success does so without acknowledging the help of others. The wise and confident acknowledge this help with gratitude.

Alfred North Whitehead

This thesis is the result of four exciting years as a PhD student. These years have been full of failures and successes—the former provided me opportunities to grow while the latter allowed me to eventually write this thesis. They have also been accompanied by many people who supported me during this journey and whom I want to thank here.

Above all, I want to express my deepest gratitude to Juliane Krämer; I could not have wished for a better advisor. I am deeply indebted to Juliane for accepting me as her first PhD student—enabling me this whole journey in the first place—and her unconditional support ever since. From identifying which of my ideas were worth pursuing and which were not, to helping me in finding new directions whenever I was lost, to providing guidance and valuable advice whenever I was in need of it: Juliane always had an open door and ear for me and all of my questions. After four years as her PhD student, I am still impressed by her genuine kindness, generosity, and inspiring optimism; seldom have I met a person as amiable as her. *Thank you, Juliane, for everything!*

I want to thank Gorjan Alagic for being a reviewer of this thesis. I am also very grateful to him for hosting me during my research visit at QuICS (Maryland) in December 2021. Gorjan was a fantastic host and I really enjoyed our long discussions about interesting research questions in front of the whiteboards as well as our online discussions since then; I look forward to continue working with him.

Special thanks to Thomas Schneider for agreeing to review this thesis, which allowed me to finish my PhD in Darmstadt. I also want to thank Christian Bischof, Rainer Hähnle, and Christian Reuter for serving on my committee.

Many thanks to my co-authors for the great and fruitful collaborations. Among them, special thanks to Jean Paul Degabriele, Tommaso Gagliardoni, Christian Janson, and Juliane Krämer for teaching me so much about doing research and writing papers.

During my time as a PhD student, I met so many wonderful people, which makes it impossible to mention everyone here. Samed Düzlü started as my office mate and became a good friend; I want to thank him for the countless discussions we had on various topics, whether it was work-related or just to kill some time in the office, and for being a great travel companion. I am very grateful to Christian Janson who always had time for a quick chat and who provided valuable advise on various occasions during my PhD. I want to thank my namesake Patrick Harasser for teaching me so many things regarding LaTeX, in particular, during the final stages of writing this thesis. Thanks to Maximilian Orlt and Olga Sanina for the many times we had a coffee and the friendship that resulted from those. I want to thank Ann-Kathrin Braun and Lucas Schabhüser for the many pub quizzes we did together—I will certainly not forget the capital of Canada. Furthermore, I want to thank Thomas Aulbach, Samed Düzlü, Juliane Krämer, Oscar Lapointe, and Michael Meyer for making my start in Regensburg such a nice experience. Finally, many thanks to all the people I did not mention here but who would have deserved it.

I was fortunate to be part of CROSSING during me PhD; not just for financial support of my research but also for the very supportive structure provided by it. For this, I want to thank Johannes Braun, Jacqueline Brendel, and Stefanie Kettler. Special thanks to Johannes Braun who answered many of my questions on countless topics. I recall that "Johannes might know that" often crossed my mind, especially in the beginning. When it came to bureaucracy, I received invaluable support from Andrea Püchner. Without her, my PhD would have probably taking much more time, as I would have failed to fill out documents properly. I am very grateful to her.

I want to thank Nina Bindel, Denise Demirel, and Lucas Schabhüser for their support during my Bachelor and Master, which encouraged me to pursue a PhD in cryptography. Special thanks also to Johannes Buchmann whose great lectures triggered my interest in cryptography in the first place. He was the one who suggested me Juliane Krämer as my advisor for which I am very grateful to him.

Thanks to my friends outside the university for distractions from work, thereby ensuring that I maintained some form of work-life balance. Especially during the beginning of the pandemic, I am grateful to Waldemar First, Andreas Muttscheller, Martin Scheuerlein, and Sascha Witascheck for the countless hours of video games together, when there was literally nothing else to do. This made a very dull period of my PhD much more bearable. Special thanks to Sascha Witascheck; although I know him for a rather short time, he is like a very old friend.

Finally, I am very grateful to my parents and sisters for always believing in me and for their unwavering support. *Thank you!*

Patrick Struck
Regensburg, April 2022

LIST OF PUBLICATIONS

Papers in Conferences and Workshops with Proceedings

[ADE+20] Nabil Alkeilani Alkadri, Poulami Das, Andreas Erwig, Sebastian Faust, Juliane Krämer, Siavash Riahi, and Patrick Struck. Deterministic wallets in a quantum world. In *CCS 2020, Virtual Event, USA*. ACM, 2020. Available as Cryptology ePrint Archive Report 2020/1149.

[BDK+20] Niklas Büscher, Daniel Demmler, Nikolaos P. Karvelas, Stefan Katzenbeisser, Juliane Krämer, Deevashwer Rathee, Thomas Schneider, and Patrick Struck. Secure two-party computation in a quantum world. In *ACNS 2020, Rome, Italy*. Springer, 2020. Available as Cryptology ePrint Archive Report 2020/411.

[DJS19] Jean Paul Degabriele, Christian Janson, and Patrick Struck. Sponges resist leakage: The case of authenticated encryption. In *ASIACRYPT 2019, Kobe, Japan*. Springer, 2019. Available as Cryptology ePrint Archive Report 2019/1034.

[GKS21] Tommaso Gagliardoni, Juliane Krämer, and Patrick Struck. Quantum indistinguishability for public key encryption. In *PQCrypto 2021, Daejeon, South Korea*. Springer, 2021. Available as Cryptology ePrint Archive Report 2020/266.

[KS20a] Juliane Krämer and Patrick Struck. Encryption schemes using random oracles: From classical to post-quantum security. In *PQCrypto 2020, Paris, France*. Springer, 2020. Available as Cryptology ePrint Archive Report 2020/129.

[KS20b] Juliane Krämer and Patrick Struck. Leakage-resilient authenticated encryption from leakage-resilient pseudorandom functions. In *COSADE 2020, Lugano, Switzerland*. Springer, 2020. Available as Cryptology ePrint Archive Report 2020/280.

[KS20c] Juliane Krämer and Patrick Struck. Security of public key encryption against resetting attacks. In *INDOCRYPT 2020, Bangalore, India*. Springer, 2020. Available as Cryptology ePrint Archive Report 2020/1316.

Preprints

[FKOS22] Sebastian Faust, Juliane Krämer, Maximilian Orlt, and Patrick Struck. On the related-key attack security of authenticated encryption schemes. 2022. Available as Cryptology ePrint Archive Report 2022/140.

[JS22] Christian Janson and Patrick Struck. Sponge-based authenticated encryption: Security against quantum attackers. 2022. Available as Cryptology ePrint Archive Report 2022/139.

CURRICULUM VITAE

since April 2022

Postdoc at the chair for Data Security and Cryptography of Prof. Dr. Juliane Krämer at University of Regensburg.

January 2022 – March 2022

PhD student at the chair for Data Security and Cryptography of Prof. Dr. Juliane Krämer at University of Regensburg.

April 2018 – December 2021

PhD student in the Athene-Young-Investigator research group QPC of Dr. Juliane Krämer at Technical University of Darmstadt.

November 2015 – March 2018

Studies (M.Sc. IT-Security) at Technical University of Darmstadt.

October 2011 – October 2015

Studies (B.Sc. Computer Science) at Technical University of Darmstadt.

ABSTRACT

The provable security paradigm is an important tool to show security of cryptographic primitives. Here, security follows from showing that an adversary cannot break a scheme with respect to some security notion. Standard security notions, however, often do not cover scenarios that might happen in practice. Examples are side-channel leakage as well as usage of keys and random coins that are somehow related. Another setting that often is not considered is security with respect to adversaries that have quantum computing power.

In this thesis we study security of schemes in advanced security notions; these notions model more sophisticated attacks which can happen when using such schemes. We develop new advanced security notions, analyse existing primitives with respect to these, and construct primitives that achieve such advanced security notions.

The first part of this thesis focuses on security outside the black-box model. Here, we develop a generic blueprint for a leakage-resilient authenticated encryption scheme from leakage-resilient functions. We then provide an instantiation entirely built from sponges. Furthermore, we provide security notions for related-key attacks against authenticated encryption schemes and analyse generic constructions with respect to these. Finally, we study the security of public key encryption schemes in case of reused random coins; we prove a simplification of the security notion which was already claimed yet backed up by a proof which was later identified as flawed.

The second part focuses on security against the glooming threat of quantum computers. First, we provide positive results for the post-quantum security of several primitives. We develop a lifting theorem for public key encryption schemes from classical proofs in the random oracle model to post-quantum proofs in the quantum random oracle model. We further show post-quantum security of the sponge-based authenticated encryption scheme developed in the first part, a generic construction for deterministic wallets, and Yao's garbled circuits. Second, we develop a quantum security notion for public key encryption schemes which allows for a quantum challenge phase; we provide both positive and negative results with respect to this security notion.

CONTENTS

1 Introduction 1

2 Background 7

I Security in Hostile Environments 41

3 Leakage-Resilient Cryptography 43

4 Misuse Security 103

II Security against Quantum Attacks 149

5 Post-Quantum Security 151

6 Quantum Security 189

7 Conclusion and Future Directions 219

List of Figures 223

List of Acronyms 227

List of Notation 231

References 233

DETAILED CONTENTS

1 Introduction **1**
1.1 Roadmap and Contribution 4

2 Background **7**
2.1 Notation 7
2.2 Primitives 8
2.3 Leakage-Resilient Cryptography 21
2.4 Related-Key Attacks 27
2.5 Related-Randomness Attacks 28
2.6 Quantum-Resistant Cryptography 30

I Security in Hostile Environments **41**

3 Leakage-Resilient Cryptography **43**
3.1 FGHF′: A Generic Construction 46
3.2 SLAE: An Instantiation based on Sponges 58
3.3 LAE from LPRF 84
3.4 Summary and Outlook 100

4 Misuse Security **103**
4.1 Related-Key Attack Security Notions 106
4.2 Related-Key Attack Security Analysis 109
4.3 Public Key Encryption Vulnerable to Resetting Attacks 128
4.4 Security Notions against Resetting Attacks 135
4.5 Summary and Outlook 147

II Security against Quantum Attacks **149**

5 Post-Quantum Security **151**
5.1 Lifting Theorem for IND-CPA Security 156
5.2 Post-Quantum Security of SLAE 163
5.3 Post-Quantum Security of Deterministic Wallets 171
5.4 Post-Quantum Security of Yao's Garbled Circuits 179
5.5 Summary and Outlook 186

6 Quantum Security **189**
6.1 Quantum Indistinguishability for PKE Schemes 194
6.2 Security Analysis for Real-World PKE Schemes 203
6.3 Classifying Other Public Key Encryption Schemes 212
6.4 Summary and Outlook 216

7 Conclusion and Future Directions **219**

List of Figures **223**

List of Acronyms **227**

List of Notation **231**

References **233**

CHAPTER 1

INTRODUCTION

Historically, cryptography was used to ensure the confidentiality of messages between two communicating parties. Famous examples are the Caesar cipher as well as the Vigenère cipher. In our modern world, cryptography is ubiquitously deployed and confidentiality is still a goal achieved by means of both asymmetric and symmetric encryption. In addition, cryptography is used nowadays for much more security goals and new cryptographic primitives were developed to meet these. Message authentication codes, for instance, ensure the authenticity and integrity of a message sent from Alice to Bob. Authenticated encryption schemes simultaneously achieve confidentiality, authenticity, and integrity of messages communicated between two parties. Digital signatures also achieve authenticity and integrity of a sent message but allow everyone to verify it—in contrast to message authentication codes which allow only the recipient to verify this. Zero-knowledge proofs allow Alice to prove knowledge of some secret information to Bob without revealing anything about this secret information. By means of multi-party computation, two[1] or more parties, holding different input values, can jointly evaluate a function on these inputs without revealing the individual inputs to the others. A crucial tool for multi-party computation is oblivious transfer which, in the simplest form, allows Alice to obliviously send one out of two messages to Bob, based on a bit known by Bob.

Security is fundamental for cryptographic primitives. This means that an adversary should not be able to break the cryptographic primitive, where a break has to be specified for the concrete scenario. For encryption schemes, for instance, a break is typically the ability to distinguish to which message a ciphertext belongs, whereas for message authentication codes it is the ability to forge a valid tag. While early design attempts were based on proposing schemes and intuitive arguments as for why they should be secure, modern cryptography typically relies on a paradigm called provable security. Here the goal is to show that under certain assumptions—like the intractability of some mathematical problem—the primitive is secure, i.e., an adversary cannot break it. The provable security paradigm provides a clearer formalism and is less error-prone than intuitive arguments

[1] In the literature the case of two parties is called secure two-party computation (STPC).

regarding security. However, the provable security paradigm is by no means free of mistakes, a prominent example is the security of OCB2 [Rog04a]. This scheme was backed up by a security proof and considered secure for more than a decade. Inoue and Minematsu identified a gap in the proof and showed an attack which breaks the authenticity of the scheme [IM18]. The attack was quickly extended to break confidentiality [Poe18] and to an attack allowing to recover plaintexts [Iwa18].[2]

To formalise what is considered a break, security in a cryptographic sense is typically defined in form of a security notion. Such notions grant an adversary certain powers, also referred to as the attack model—something that an adversary is expected to be able to do in the real world—and a goal—which describes what is considered a break of the primitive. A basic security notion for public key encryption schemes is called ciphertext indistinguishability under chosen plaintext attacks (IND-CPA), where IND corresponds to the goal while CPA corresponds to the attack model. Here, the adversary, henceforth also referred to as Eve, gets the public key. This corresponds to the power that the adversary can encrypt arbitrary messages of her choice. As the name suggests, this key is public so it is reasonable that the adversary knows the key. The goal of the adversary Eve is to distinguish whether a ciphertext c is the encryption of a message m_0 or a message m_1, where both m_0 and m_1 are chosen by Eve. The reason for letting Eve choose the messages is twofold. First, it reflects prior knowledge that Eve has about the communication between Alice and Bob. For instance, if Alice would send either an encryption of "Yes" or an encryption of "No" to Bob. Second, it guarantees security even in the worst-case scenario. One can think of an encryption scheme for which ciphertexts of certain messages exhibit a special structure, which easily distinguishes them from other ciphertexts. This scheme would not be secure as there exists an adversary that picks these easy-to-distinguish messages which allows to determine the message that was encrypted.

A security proof then shows, under certain assumptions, that no efficient[3] adversary exists that can break the primitive according to that particular security notion. A common technique to show this is by means of a reduction[4]. A reduction is an algorithm that transforms an adversary, which breaks a primitive with respect to some security notion, into an algorithm that efficiently solves some mathematical problem or hardness assumption. Then, by the assumption that the mathematical problem is intractable to solve, this rules out the existence of the hypothetical adversary breaking the primitive.

However, a security proof only guarantees security against adversaries in the concrete model; it does not reveal anything about adversaries outside this model. Coming back to the example of IND-CPA this means for instance that an IND-CPA-secure scheme might be completely broken if the adversary can somehow obtain the decryption of ciphertexts.[5] In the real world, Eve could simply pick a ciphertext c and send it to Bob. Based on Bobs behaviour she can learn what the message was. A simple scenario [KL20] is an e-mail communication where Eve sends a message in an encrypted form to Bob whose response

[2] These results were later published jointly in [IIMP19].

[3] Efficiency turns out to be crucial here as brute-force attacks are always possible. However, breaking a scheme after, say, thousands of years is typically not considered a security problem.

[4] Note that other proof techniques exist [Lin17, Can01].

[5] In fact, most schemes are broken if the adversary can obtain the decryption of ciphertexts. Security is then achieved via generic transformations that enhance the scheme by using additional primitives.

will quote the message. By sending an arbitrary ciphertext, Bob will effectively act as a decryption oracle for Eve. To cover this stronger setting, the security notion ciphertext indistinguishability against chosen ciphertext attacks (IND-CCA) was developed. It is similar to IND-CPA as it shares the same goal (IND) but the stronger attack model (CCA) that grants Eve additionally access to a decryption oracle which decrypts arbitrary ciphertexts[6] for her.

The above notions, both IND-CPA and IND-CCA, implicitly assume that Eve interacts in a black-box manner with the primitive, i.e., she only gets access to the input/output behaviour but not to the internal workings. This assumption can cease to hold in practice. In case of a simple implementation of the RSA encryption scheme, the runtime during decryption depends on the secret key; for each bit that is 0, an exponentiation is performed while for each bit that is 1, an exponentiation as well as a multiplication is performed. Kocher [Koc96] showed that measuring the runtime allows to derive information about the secret key. Even though the scheme might be secure in a black-box setting[7], it can easily be attacked in practice via such side-channel attacks unless proper countermeasures are deployed. Another implicit assumption is that good randomness is available as the challenge ciphertexts are generated using fresh random coins per encryption. Ristenpart and Yilek [RY10] showed that this can also fail in practice. Precisely, they show that snapshots of virtual machines can be exploited to reuse random coins. Since the random coins are derived from the current state of the machine, e.g., the current state of the RAM, a snapshot of this state would derive the same random coins. A proof with respect to either IND-CPA or IND-CCA, again, does not reveal anything about the security when an adversary can force the reuse of random coins.

Another point is the computational power of the adversary. Adversaries with access to a quantum computer can leverage Shor's algorithm [Sho94] to break public key cryptography based on number-theoretic assumptions such as RSA. This requires a transition to post-quantum cryptography—primitives that can be executed on classical computers and are based on problems that are assumed to withstand quantum attackers. Again, security notions and security proofs that do not cover the additional quantum power of an adversary will cease to be relevant once large-scale quantum computers exist. The most prominent example are security notions in the random oracle model (ROM) [BR93]. This model idealises primitives like hash functions in order to facilitate a security proof. About a decade ago, Boneh et al. [BDF+11] showed that the random oracle model is inadequate when considering adversaries with quantum computing power and advocated the stronger model called quantum random oracle model (QROM). This model grants the adversary quantum access to random oracles, which is justified by the simple observation that hash functions—which are replaced by random oracles in the proof—are public. Hence, a quantum adversary can implement and evaluate it on its quantum computer. Despite this, primitives are still often only analysed in the ROM which would only ensure security if the adversary refrains from using its quantum computing power with respect to the

[6] This notion requires the obvious restriction that Eve must not simply ask for a decryption of the challenge ciphertext which turns out to have subtle difficulties that go way beyond the scope of this motivation [BHK15].

[7] Note that, unlike the Rabin encryption scheme [Rab79], the RSA encryption scheme is not backed up by a security proof.

hash function, an assumption that is not realistic. Another problem might occur when the adversary gets quantum access to the primitive. While this is currently considered more of a theoretic threat, it is ill-advised to simply ignore this. The simple reason is that it is currently not clear how exactly quantum computers will affect cryptography, in particular, what types of attacks are possible; attacks that might seem artificial or unrealistic nowadays might turn out to be realistic in the future.

This thesis focuses on advanced security notions that model more sophisticated attacks that can happen in practice. We develop new advanced security notions, analyse existing primitives with respect to these, and construct primitives that achieve such advanced security notions.

1.1 Roadmap and Contribution

Below we provide a roadmap of this thesis.

Chapter 2 (Background): This chapter provides the necessary background for the work. We recall existing security notions, cryptographic primitives, and results.

Part I (Security in Hostile Environments): The first part addresses security outside the black-box model and it consists of two chapters. The former considers the setting of leakage-resilient cryptography which models side-channel leakage while the latter considers misuse security of encryption schemes as a result of related-key attacks and related-randomness attacks.

Chapter 3 (Leakage-Resilient Cryptography): In this chapter, we develop a generic blueprint for leakage-resilient AEAD schemes which we call the FGHF′ construction. It constructs an authenticated encryption scheme with associated data (AEAD) from two fixed-input length functions $\mathcal{F}$ and $\mathcal{F}'$ as well as a pseudorandom generator $\mathcal{G}$ and a vector hash function $\mathcal{H}$. We show that leakage resilience of the AEAD scheme reduces to the leakage resilience of the functions $\mathcal{F}$ and $\mathcal{F}'$. We further provide SLAE, an instantiation of the FGHF′ construction entirely based on sponges. It consists of a fixed-input length function SLFUNC which is used to instantiate $\mathcal{F}$ and $\mathcal{F}'$, a pseudorandom generator SPRG to instantiate $\mathcal{G}$, and a vector hash function SVHASH to instantiate $\mathcal{H}$.

The results presented in Section 3.1 and Section 3.2 are based on a paper that appeared at ASIACRYPT'19 [DJS19]. They are joint work together with Jean Paul Degabriele and Christian Janson, except for Theorem 3.2.1 and Theorem 3.2.2 which were contributed by Jean Paul Degabriele. The improvements of the FGHF′ construction, presented in Section 3.3, appeared at COSADE'20 [KS20b]. They were developed in joint work with Juliane Krämer, where I contributed the idea for the project.

Chapter 4 (Misuse Security): In this chapter, we develop security notions for authenticated encryption schemes against related-key attacks and analyse generic

constructions for authenticated encryption schemes with respect to these notions, giving both positive and negative results. We further analyse the encryption scheme and message authentication code underlying the FGHF′ construction and show that its related-key attack security reduces to the related-key attack security of the underlying functions. This part is based on research that I conducted together with Sebastian Faust, Juliane Krämer, and Maximilian Orlt which is currently under submission [FKOS22]. The initial project idea to study the related-key attack security of authenticated encryption schemes was contributed by me.

Furthermore, we show that security notions for resetting attacks, a specific case of related-randomness attacks, can be simplified to a single challenge as is typical for security notions considering public key encryption. This simplification was already claimed to hold for resetting attacks, however, the proof was later identified to be flawed. We show an inherent problem of the flawed proof and give a different proof to nevertheless confirm the claim. Additionally, we show the strength of resetting attacks by defining a class of public key encryption schemes that is vulnerable against these attacks and show that several real-world schemes lie in this class. These results were published at INDOCRYPT'20 [KS20c] which I co-authored together with Juliane Krämer. The idea for the project was contributed by me.

Part II (Security against Quantum Attacks): The second part addresses security against the glooming threat of quantum computers. It consists of two chapters. The former addresses the scenario of post-quantum security where adversaries have local quantum computing power. The latter addresses the stronger scenario of quantum security where adversaries get quantum access to the cryptographic primitives.

Chapter 5 (Post-Quantum Security): In this chapter, we develop a lifting theorem for post-quantum security of public key encryption schemes based on random oracles. We use this lifting theorem to prove several public key encryption schemes secure in the QROM which so far were only proven secure in the ROM. The results are based on a paper that appeared at PQCrypto'20 [KS20a], which I developed jointly with Juliane Krämer.

We further show that the post-quantum security of the generic FGHF′ construction, developed in Chapter 3, boils down to the post-quantum security of the underlying components. We then prove that the sponge-based instantiations are post-quantum secure, which yields the post-quantum security of the authenticated encryption scheme Slae. These results are based on joint research with Christian Janson that is currently under submission [JS22], where the initial idea was contributed by me.

Finally, we provide a post-quantum security proof for a generic construction of deterministic wallets and show that Yao's garbled circuits are post-quantum secure if the underlying building blocks are. The former is based on a paper that appeared at CCS'20 [ADE+20] which is based on a collaboration with Nabil Alkailani Alkadri, Poulami Das, Andreas Erwig, Sebastian Faust, Juliane

Krämer, and Siavash Riahi. The latter is based on a paper that appeared at ACNS'20 [BDK+20] which was joint work with Niklas Büscher, Daniel Demmler, Nikolaos P. Karvelas, Stefan Katzenbeisser, Juliane Krämer, Deevashwer Rathee, and Thomas Schneider. For both works, the results presented here were contributed by Juliane Krämer and me.

Chapter 6 (Quantum Security): In this chapter, we develop a quantum security notion for public key encryption schemes which allows for a quantum indistinguishability phase. This solves definitional issues identified nine years ago by Boneh and Zhandry. We further provide positive and negative results for public key encryption schemes with respect to this notion and initiate the study of classifying public key encryption schemes in view of applicability of the quantum security notion.

The results are based on research that appeared at PQCrypto'21 [GKS21] which I co-authored together with Tommaso Gagliardoni and Juliane Krämer. The idea for this project was contributed by me.

Chapter 7 (Conclusion): This chapter concludes the thesis and gives an outlook for further research directions.

CHAPTER 2

BACKGROUND

Contents

2.1 Notation **7**
2.2 Primitives **8**
2.3 Leakage-Resilient Cryptography **21**
2.4 Related-Key Attacks **27**
2.5 Related-Randomness Attacks **28**
2.6 Quantum-Resistant Cryptography **30**

In this section we give the necessary background for this thesis. We start with the notation and the definition of cryptographic primitives in Section 2.1 and Section 2.2, respectively. The background for leakage-resilient cryptography is given in Section 2.3. Section 2.4 covers the background related-key attacks while the background for related-randomness attacks is given in Section 2.5. Finally, Section 2.6 covers the background on quantum-resistant cryptography.

2.1 Notation

For an integer k, we write $[k]$ for the set $\{1, \ldots, k\}$. Let X be a bit string and y be an integer, then $\lceil X \rceil^y$ denotes the leftmost y bits of X; similarly, $\lfloor X \rfloor_y$ denotes the rightmost y bits of X. The empty string is denoted as ε. We write $x \leftarrow y$ to denote that x is assigned to be y. For a set $\mathcal{S}$, we write $x \leftarrow\!\!\$\, \mathcal{S}$ to denote that an element from $\mathcal{S}$ is chosen at random and assigned to x. We further abbreviate $\mathcal{S} \leftarrow \mathcal{S} \cup \{s\}$ by $\mathcal{S} \leftarrow\!\!\cup\, s$. For sets $\mathcal{X}$ and $\mathcal{Y}$, the set of all function from set of functions from $\mathcal{X}$ to $\mathcal{Y}$ is denoted by $\mathsf{Func}(\mathcal{X}, \mathcal{Y})$ as well as $\mathcal{Y}^{\mathcal{X}}$. When considering keyed functions from $\mathcal{X}$ to $\mathcal{Y}$, with key space $\mathcal{K}$, we write $\mathsf{Func}(\mathcal{K}, \mathcal{X}, \mathcal{Y})$.

We use game-based proofs [BR06, Sho04], where an adversary, typically denoted as $\mathcal{A}$, plays a game. Unless mentioned otherwise, adversaries are considered to be efficient algorithms, where efficient means polynomial-time in the security parameter. For adversaries that can query oracles, it is understood that the number of queries are polynomial in the security parameter. For a game G and an adversary $\mathcal{A}$, we write $\mathsf{G}^{\mathcal{A}} \Rightarrow y$ if the game outputs y when played by $\mathcal{A}$. In our case, the game output will either be 1 or 0, indicating whether the adversary has won or lost the game. Analogously, we write $\mathcal{A}^{\mathsf{G}} \Rightarrow y$ to denote that $\mathcal{A}$ outputs y when playing game G. We mainly use distinguishing games in which the adversary has to guess a randomly chosen bit b. To scale the advantage of an adversary $\mathcal{A}$ to the interval from 0 to 1, its advantage in a distinguishing game G is defined as

$$\mathbf{Adv}^{\mathsf{G}}(\mathcal{A}) = \left|2 \Pr[\mathsf{G}^{\mathcal{A}} \Rightarrow 1] - 1\right| .$$

Reformulation to adversarial advantage yields

$$\begin{aligned}
\mathbf{Adv}^{\mathsf{G}}(\mathcal{A}) &= \left|2 \Pr[\mathsf{G}^{\mathcal{A}} \Rightarrow 1] - 1\right| \\
&= \left|2 \left(\Pr[\mathsf{G}^{\mathcal{A}} \Rightarrow 1 \cap b = 0] + \Pr[\mathsf{G}^{\mathcal{A}} \Rightarrow 1 \cap b = 1]\right) - 1\right| \\
&= \left|2 \left(\Pr[\mathsf{G}^{\mathcal{A}} \Rightarrow 1 \,|\, b = 0] \Pr[b = 0] + \Pr[\mathsf{G}^{\mathcal{A}} \Rightarrow 1 \,|\, b = 1] \Pr[b = 1]\right) - 1\right| \\
&= \left|\Pr[\mathsf{G}^{\mathcal{A}} \Rightarrow 1 \,|\, b = 0] + \Pr[\mathsf{G}^{\mathcal{A}} \Rightarrow 1 \,|\, b = 1] - 1\right| \\
&= \left|\Pr[\mathcal{A}^{\mathsf{G}} \Rightarrow 0 \,|\, b = 0] + \Pr[\mathcal{A}^{\mathsf{G}} \Rightarrow 1 \,|\, b = 1] - 1\right| \\
&= \left|\Pr[\mathcal{A}^{\mathsf{G}} \Rightarrow 0 \,|\, b = 0] - \Pr[\mathcal{A}^{\mathsf{G}} \Rightarrow 0 \,|\, b = 1]\right| .
\end{aligned}$$

Analogously, we get

$$\begin{aligned}
\mathbf{Adv}^{\mathsf{G}}(\mathcal{A}) &= \left|2 \Pr[\mathsf{G}^{\mathcal{A}} \Rightarrow 1] - 1\right| \\
&= \left|2 \left(\Pr[\mathsf{G}^{\mathcal{A}} \Rightarrow 1 \cap b = 1] + \Pr[\mathsf{G}^{\mathcal{A}} \Rightarrow 1 \cap b = 0]\right) - 1\right| \\
&= \left|2 \left(\Pr[\mathsf{G}^{\mathcal{A}} \Rightarrow 1 \,|\, b = 1] \Pr[b = 1] + \Pr[\mathsf{G}^{\mathcal{A}} \Rightarrow 1 \,|\, b = 0] \Pr[b = 0]\right) - 1\right| \\
&= \left|\Pr[\mathsf{G}^{\mathcal{A}} \Rightarrow 1 \,|\, b = 1] + \Pr[\mathsf{G}^{\mathcal{A}} \Rightarrow 1 \,|\, b = 0] - 1\right| \\
&= \left|\Pr[\mathcal{A}^{\mathsf{G}} \Rightarrow 1 \,|\, b = 1] + \Pr[\mathcal{A}^{\mathsf{G}} \Rightarrow 0 \,|\, b = 0] - 1\right| \\
&= \left|\Pr[\mathcal{A}^{\mathsf{G}} \Rightarrow 1 \,|\, b = 1] - \Pr[\mathcal{A}^{\mathsf{G}} \Rightarrow 1 \,|\, b = 0]\right| .
\end{aligned}$$

Hence we bound the advantage by bounding the difference of the adversary in outputting a bit (either 1 or 0) for the two possible cases of the secret bit b.

2.2 Primitives

We recall the definition of several cryptographic primitives and basic security notions.

2.2.1 Message Authentication Codes

A message authentication code (MAC) is a symmetric primitive allowing two parties, which share a secret key K, to send authenticated messages to one another. It consists

of a tagging algorithm to generate a tag for a message and a verification algorithm which verifies if a tag belongs to a message or not.

Definition 2.2.1. *A message authentication code* $\text{MAC} = (\texttt{Tag}, \texttt{Ver})$ *is a pair of efficient algorithms associated with key space* $\mathcal{K}$ *and domain space* $\mathcal{X}$ *such that:*

- *The deterministic tagging algorithm* $\texttt{Tag}\colon \mathcal{K} \times \mathcal{X} \to \{0,1\}^\tau$ *takes as input a key* K *and an element* X*. It returns a tag* T *of size* $\{0,1\}^\tau$*.*
- *The deterministic verification algorithm* $\texttt{Ver}\colon \mathcal{K} \times \mathcal{X} \times \{0,1\}^\tau \to \{0,1\}$ *takes as input a key* K*, an element* X*, and a tag* T *and outputs* 1 *indicating that the input is valid, or otherwise* 0*.*

We say that a MAC scheme is correct*, if for all* $K \in \mathcal{K}$ *and any admissible input* $X \in \mathcal{X}$*, it holds that* $\texttt{Ver}(K, X, \texttt{Tag}(K, X)) = 1$*.*

A typical approach for MACs are canonical MACs. These are MACs where the verification algorithm recomputes the tag for the message and accepts if it equals the given tag and rejects otherwise.

The standard security notion for MACs is *strong unforgeability under chosen message attacks* (SUF-CMA). Intuitively, it means that an adversary, after seeing tags on messages of its choice, can not forge a tag—not even forge a different tag for a message for which it has already seen a tag. Below we formally define SUF-CMA security for a MAC.

Definition 2.2.2. *Let* MAC *be a message authentication code. We define the* SUF-CMA *advantage of an adversary* $\mathcal{A}$ *making at most* q_T *queries to its tag oracle and* q_F *many queries to its forge oracle as*

$$\mathbf{Adv}^{\mathsf{SUFCMA}}_{\text{MAC}}(\mathcal{A}) \coloneqq \Pr[\mathsf{SUFCMA}^{\mathcal{A}} \Rightarrow 1]\,,$$

where the respective game is depicted in Figure 2.1.

Game SUF-CMA	**oracle** $\mathsf{Tag}(K, X)$	**oracle** $\mathsf{Forge}(K, X, T)$
$K \leftarrow\!\!\$\, \mathcal{K}$	$T \leftarrow \texttt{Tag}(K, X)$	**if** $(X, T) \in \mathcal{S}$
$\mathcal{S} \leftarrow \emptyset$	$\mathcal{S} \leftarrow\!\!\cup\, (X, T)$	**return** $\bot$
$\mathsf{win} \leftarrow 0$	**return** T	$V \leftarrow \texttt{Ver}(K, X, T)$
$\mathcal{A}^{\mathsf{Tag}(K,\cdot),\mathsf{Forge}(K,\cdot,\cdot)}$		**if** $V = \top$
return win		$\mathsf{win} \leftarrow 1$
		return V

Figure 2.1: Security game SUFCMA.

A weaker variant of SUF-CMA is *existential unforgeability under chosen message attack* (EUF-CMA) which requires a forged tag for a message the adversary has not received a tag for. In case of deterministic MACs—where every message has a unique tag—the two notions are, in fact, equivalent.

2.2.2 Symmetric Encryption

We specify encryption schemes as deterministic algorithms using an initialization vector to randomise ciphertexts. We generally consider the case where the initialization vector is a nonce, i.e., it never repeats, for instance, by using a counter. Nonce-based encryption was formalised by Rogaway [Rog04b], but already appeared earlier [Rog02, RBBK01].

Definition 2.2.3. *A symmetric (key) encryption scheme* $\text{SE} = (\texttt{Enc}, \texttt{Dec})$ *is a pair of efficient algorithms such that:*

- *The deterministic encryption algorithm* $\texttt{Enc}\colon \mathcal{K} \times \mathcal{N} \times \mathcal{M} \to \mathcal{C}$ *takes as input a key* K*, a nonce* N*, and a message* M*. It outputs a ciphertext* C*.*
- *The deterministic decryption algorithm* $\texttt{Dec}\colon \mathcal{K} \times \mathcal{N} \times \mathcal{C} \to \mathcal{M}$ *takes as input a key* K*, a nonce* N*, and a ciphertext* C *and outputs a message* M*.*

The typical security notion for symmetric encryption schemes is *ciphertext indistinguishability under chosen plaintext attacks* (IND-CPA). This notion grants the adversary access to an oracle which can be queried on two messages and encrypts either the left or the right message. The goal of the adversary is to determine which of the messages was encrypted.

Definition 2.2.4 (IND-CPA Security)**.** *Let* $\text{SE} = (\texttt{Enc}, \texttt{Dec})$ *be a symmetric encryption scheme and the game* INDCPA *be as defined in Figure 2.2. For any nonce-respecting adversary* $\mathcal{A}$*, its corresponding* IND-CPA *advantage is given by*

$$\mathbf{Adv}^{\mathsf{INDCPA}}_{\text{SE}}(\mathcal{A}) \coloneqq \left|2 \Pr[\mathsf{INDCPA}^{\mathcal{A}} \Rightarrow 1] - 1\right| .$$

Game INDCPA	**oracle** LR-Enc(N, M_0, M_1)	**oracle** Enc(N, M)
$b \leftarrow\!\!\$\ \{0,1\}$	$C \leftarrow \texttt{Enc}(K, N, M_b)$	$C \leftarrow \texttt{Enc}(K, N, M)$
$K \leftarrow\!\!\$\ \mathcal{K}$	**return** C	**return** C
$b' \leftarrow \mathcal{A}^{\mathsf{LR\text{-}Enc},\mathsf{Enc}}()$		
return $(b' = b)$		

Figure 2.2: Security game INDCPA.

The above notion is of the form left-or-right which allows the adversary to specify two messages. Another variant of the notion is real-or-random in which the adversary only queries one message and the oracle either encrypts this message or a message picked uniformly at random. Equivalence between these variants is shown in [BDJR97]. There is also the stronger notion [RBBK01] which we will call real-or-ideal. In this notion the adversary still only queries one message but instead of encrypting a random message, as done in the real-or-random case, the oracle will return a random bit string of appropriate length. While this variant is strictly stronger than the other two[8], they are all considered sufficient for practice.

[8] A simple separation is an encryption scheme where every ciphertext has some dummy bit prepended which is 0. Whenever the adversary receives a ciphertext starting with a 1, it immediately knows that it is a random bit string rather than the encryption of the queried message.

2.2.3 Authenticated Encryption with Associated Data

Authenticated encryption with associated data (AEAD) was formalised in [Rog02]. It is used to achieve both confidentiality and authenticity of messages communicated between two parties which share some pre-shared key. Besides the actual message, there is also associated data which corresponds to additional information that needs to be authenticated but not to be confidential, a typical examples are packet headers for routing information.

Definition 2.2.5. *An* authenticated encryption scheme with associated data (AEAD) $\textsc{Aead} = (\mathtt{Enc}, \mathtt{Dec})$ *is a pair of efficient algorithms associated with key space $\mathcal{K}$, nonce space $\mathcal{N}$, associated-data space $\mathcal{H}$, message space $\mathcal{M}$, and ciphertext space $\mathcal{C}$ such that:*

- *The deterministic encryption algorithm* $\mathtt{Enc}\colon \mathcal{K} \times \mathcal{N} \times \mathcal{H} \times \mathcal{M} \to \mathcal{C}$ *takes as input a secret key K, a nonce N, associated data A, and a message M. It outputs a ciphertext C.*
- *The deterministic decryption algorithm* $\mathtt{Dec}\colon \mathcal{K} \times \mathcal{N} \times \mathcal{H} \times \mathcal{C} \to \mathcal{M} \cup \{\perp\}$ *takes as input a secret key K, a nonce N, associated data A, and a ciphertext C. It outputs a message M or $\perp$ indicating an invalid ciphertext.*

We say that an AEAD scheme is correct*, if for all $K \in \mathcal{K}$, $N \in \mathcal{N}$, $A \in \mathcal{H}$ and $M \in \mathcal{M}$, it holds that* $\mathtt{Dec}(K, N, A, \mathtt{Enc}(K, N, A, M)) = M$*. It is called* tidy *if for any $K \in \mathcal{K}$, any $N \in \mathcal{N}$, any associated data $A \in \mathcal{H}$, any $M \in \mathcal{M}$, and any $C \in \mathcal{C}$ with* $\mathtt{Dec}(K, N, A, C) = M$*, it holds that* $\mathtt{Enc}(K, N, A, M) = C$*.*

The sets $\mathcal{K}$, $\mathcal{N}$, $\mathcal{H}$, $\mathcal{M}$, and $\mathcal{C}$, denote the key space, nonce space, associated data space, message space, and ciphertext space, respectively. Throughout this work, we consider $\mathcal{K} = \{0,1\}^k$, $\mathcal{N} = \{0,1\}^\nu$, $\mathcal{H} = \{0,1\}^\alpha$, $\mathcal{M} = \{0,1\}^*$, and $\mathcal{C} = \{0,1\}^*$. We will generally use μ and γ to denote the length of a message and ciphertext, respectively.

Security for an AEAD scheme is defined by a game in which the adversary has access to two oracles. The former either implements the real encryption algorithm or returns random bit strings. The latter either implements the real decryption algorithm or simply rejects any queried ciphertext. The goal of the adversary is to distinguish these cases. Below we give the formal definition.

Definition 2.2.6 (AE Security)**.** *Let* $\textsc{Aead} = (\mathtt{Enc}, \mathtt{Dec})$ *be an authenticated encryption scheme with associated data and the game* AE *be as defined in Figure 2.3. For any nonce-respecting adversary $\mathcal{A}$ that never forwards or repeats queries to or from the oracles* Enc *and* Dec*, its corresponding* AE *advantage is given by*

$$\mathbf{Adv}^{\mathsf{AE}}_{\textsc{Aead}}(\mathcal{A}) \coloneqq \left|2 \Pr[\mathsf{AE}^{\mathcal{A}} \Rightarrow 1] - 1\right| .$$

Besides the unified definition stated above, security for AEAD schemes can equivalently be defined as achieving both *ciphertext indistinguishability under chosen plaintext attacks* (IND-CPA) and *integrity of ciphertexts* (INT-CTXT) security. The former (IND-CPA) is the same as for symmetric encryption schemes except adapted to the syntax of AEAD schemes, i.e., allowing associated data. The security game is displayed in Figure 2.4. The latter (INT-CTXT) exhibits similarities to SUF-CMA security for MACs. It grants the

Game AE	**oracle** $\mathsf{Enc}(N, A, M)$	**oracle** $\mathsf{Dec}(N, A, C)$
$b \leftarrow_\$ \{0,1\}$	$C \leftarrow \mathtt{Enc}(K, N, A, M)$	**if** $b = 0$
$K \leftarrow_\$ \mathcal{K}$	**if** $b = 0$	**return** $\bot$
$b' \leftarrow \mathcal{A}^{\mathsf{Enc},\mathsf{Dec}}()$	**return** $C' \leftarrow_\$ \{0,1\}^{\lvert C\rvert}$	$M \leftarrow \mathtt{Dec}(K, N, A, C)$
return $(b' = b)$	**else**	**return** M
	return C	

Figure 2.3: Security game AE.

Game INDCPA	**oracle** LR-$\mathsf{Enc}(N, A, M_0, M_1)$	**oracle** $\mathsf{Enc}(N, A, M)$
$b \leftarrow_\$ \{0,1\}$	$C \leftarrow \mathtt{Enc}(K, N, A, M_b)$	$C \leftarrow \mathtt{Enc}(K, N, A, M)$
$K \leftarrow_\$ \mathcal{K}$	**return** C	**return** C
$b' \leftarrow \mathcal{A}^{\mathsf{LR\text{-}Enc},\mathsf{Enc}}()$		
return $(b' = b)$		

Figure 2.4: Security game INDCPA.

Game INTCTXT	**oracle** $\mathsf{Enc}(N, A, M)$	**oracle** $\mathsf{Forge}(N, A, C)$
$K \leftarrow_\$ \mathcal{K}$	**if** $(N, \cdot, \cdot) \in \mathcal{S}$	**if** $(N, A, C) \in \mathcal{S}$ **then**
$\mathcal{S} \leftarrow \emptyset$	**return** $\bot$	**return** $\bot$
$\mathsf{win} \leftarrow 0$	$C \leftarrow \mathtt{Enc}(K, N, A, M)$	$d \leftarrow \mathtt{Dec}(K, N, A, C)$
$\mathcal{A}^{\mathsf{Enc},\mathsf{Forge}}()$	$\mathcal{S} \leftarrow_\cup (N, A, C)$	**if** $d \neq \bot$
return win	**return** C	$\mathsf{win} \leftarrow 1$
		return d

Figure 2.5: Security game INTCTXT.

adversary access to an oracle which encrypts arbitrary messages for the adversary. To win the game the adversary has to forge a valid ciphertext, i.e., one that decrypts to anything different than $\perp$. The security game is displayed in Figure 2.5. Equivalence between the unified notion and the case of separate notions has been shown by Shrimpton [Shr04] and we skip the formal definition with respect to separate notions here.

N-Schemes

AEAD schemes can be constructed by combining a symmetric encryption scheme and a MAC. Bellare and Namprempre analysed three generic composition paradigms for this:

1. Encrypt-and-MAC (E&M). The message M gets encrypted using the encryption scheme, yielding a ciphertext C_e and a tag is computed for the message M using the MAC, yielding a tag T. The ciphertext C of the AEAD scheme consists of the concatenation of C_e and T.

2. Encrypt-then-MAC (EtM). The message M gets encrypted using the encryption scheme, yielding a ciphertext C_e and a tag is computed for the ciphertext C_e using the MAC, yielding a tag T. The ciphertext C of the AEAD scheme consists of the concatenation of C_e and T.

3. MAC-then-Encrypt (MtE). A tag T of the message M is computed using the MAC. The ciphertext C of the AEAD then equals the ciphertext obtained by encrypting the concatenation of the message M and the tag T.

Bellare and Namprempre showed that only Encrypt-then-MAC (EtM), out of the three paradigms, yields a secure AEAD scheme [BN00, BN08]. However, Namprempre et al. [NRS14] analysed different constructions following these paradigms. Among other schemes, they provide 20 so-called N-schemes. For N1, N2, and N3 which follow the composition paradigms E&M, EtM, and MtE, respectively, they prove security based on the underlying encryption scheme and MAC. The scheme N4 was a special case which was neither proven secure nor proven insecure; later, Berti et al. showed that N4 is not secure in general but can achieve security if the underlying encryption satisfies an additional security property [BPP18]. For the other N-schemes, Namprempre et al. either provided attacks or argued that they do not satisfy the syntax of AEAD schemes.

In this work we study the three AEAD schemes N1, N2, and N3, due to Namprempre et al. [NRS14], which are visualised in Figure 2.6, Figure 2.7, and Figure 2.8, respectively. The pseudocode for each of the schemes is given in Figure 2.9.

2.2.4 Public Key Encryption

In this section we give the formal definition for public key encryption schemes and the correctness of such schemes.

Definition 2.2.7. *A* public key encryption (PKE) scheme *is a tuple* $(\texttt{KGen}, \texttt{Enc}, \texttt{Dec})$ *of three efficient algorithms such that:*

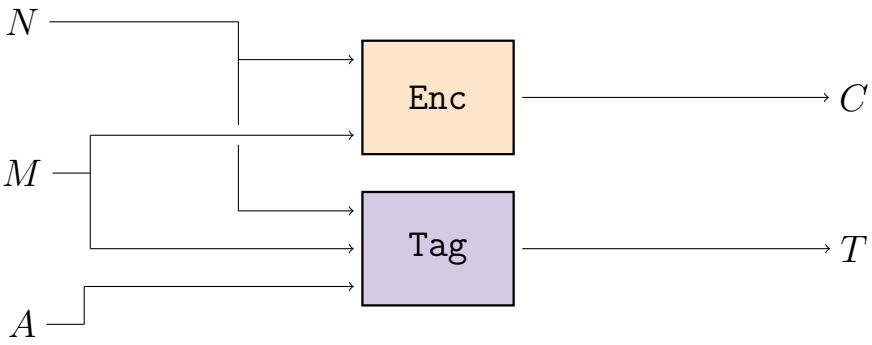

Figure 2.6: Visualisation of the AEAD scheme N1 [NRS14].

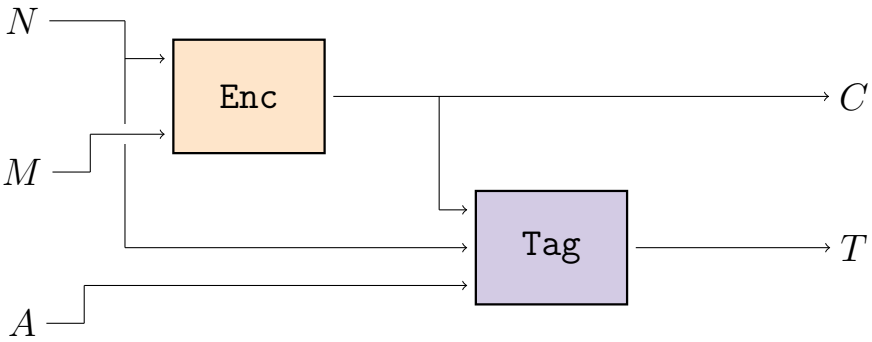

Figure 2.7: Visualisation of the AEAD scheme N2 [NRS14].

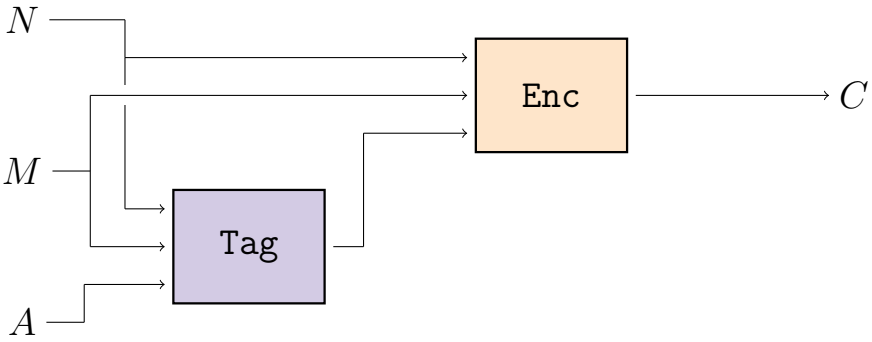

Figure 2.8: Visualisation of the AEAD scheme N3 [NRS14].

$\texttt{Enc}(K, N, A, M)$ in N1

parse K **as** (K_e, K_m)
$C_e \leftarrow \texttt{Enc}^S(K_e, N, M)$
$T \leftarrow \texttt{Tag}(K_m, N, A, M)$
$C \leftarrow C_e \parallel T$
return C

$\texttt{Enc}(K, N, A, M)$ in N2

parse K **as** (K_e, K_m)
$C_e \leftarrow \texttt{Enc}^S(K_e, N, M)$
$T \leftarrow \texttt{Tag}(K_m, N, A, C_e)$
$C \leftarrow C_e \parallel T$
return C

$\texttt{Enc}(K, N, A, M)$ in N3

parse K **as** (K_e, K_m)
$T \leftarrow \texttt{Tag}(K_m, N, A, M)$
$C \leftarrow \texttt{Enc}^S(K_e, N, M \parallel T)$
return C

$\texttt{Dec}(K, N, A, C)$ in N1

parse K **as** (K_e, K_m)
parse C **as** (C_e, T_m)
$M \leftarrow \texttt{Dec}^S(K_e, N, C_e)$
if $\texttt{Ver}(K_m, N, A, M, T) = 0$
  return $\perp$
return M

$\texttt{Dec}(K, N, A, C)$ in N2

parse K **as** (K_e, K_m)
parse C **as** (C_e, T_m)
if $\texttt{Ver}(K_m, N, A, C_e, T) = 0$
  return $\perp$
$M \leftarrow \texttt{Dec}^S(K_e, N, C_e)$
return M

$\texttt{Dec}(K, N, A, C)$ in N3

parse K **as** (K_e, K_m)
$M \parallel T \leftarrow \texttt{Dec}^S(K_e, N, C)$
if $\texttt{Ver}(K_m, N, A, M, T) = 0$
  return $\perp$
return M

Figure 2.9: Pseudocode of N1 (left), N2 (middle), and N3 (right) in terms of a symmetric encryption scheme $(\texttt{Enc}^S, \texttt{Dec}^S)$ and a MAC $(\texttt{Tag}, \texttt{Ver})$.

- $\texttt{KGen}\colon \mathbb{N} \times \mathcal{R} \to \mathcal{PK} \times \mathcal{SK}$ *is the key generation algorithm which takes a security parameter* λ *and a randomness* r *as input, and returns a keypair consisting of a public key* pk *and a secret key* sk*. If clear from the context, we will denote it by* $(pk, sk) \leftarrow\!\!\$\ \texttt{KGen}()$*. We will generally drop the security parameter.*
- $\texttt{Enc}\colon \mathcal{PK} \times \mathcal{M} \times \mathcal{R} \to \mathcal{C}$ *is the encryption algorithm which takes a public key* pk*, a message* m*, and a randomness* r *as input, and returns a ciphertext* c*. It will be usually denoted by* $c \leftarrow\!\!\$\ \texttt{Enc}_{pk}(m)$ *or* $c \leftarrow \texttt{Enc}_{pk}(m; r)$*.*
- $\texttt{Dec}\colon \mathcal{SK} \times \mathcal{C} \to \mathcal{M}$ *is the (deterministic) decryption algorithm*[9] *which takes as input a secret key* sk *and a ciphertext* c*, and returns a message* m*. It will be usually denoted by* $m \leftarrow \texttt{Dec}_{sk}(c)$*.*

By $\mathcal{PK}$*,* $\mathcal{SK}$*,* $\mathcal{M}$*,* $\mathcal{C}$*, and* $\mathcal{R}$*, we denote the public key space, secret key space, message space, ciphertext space, and randomness space, respectively.*

We assume without loss of generality that the randomness space for key generation and encryption are identical. Below we define two notions of correctness for PKE schemes.

Definition 2.2.8 (Perfectly Correct PKE)**.** *A PKE scheme* $\textsc{Pke} = (\texttt{KGen}, \texttt{Enc}, \texttt{Dec})$ *is* perfectly correct *if for any* $(pk, sk) \leftarrow\!\!\$\ \texttt{KGen}()$*,* $m \in \mathcal{M}$*, and* $r \in \mathcal{R}$*, it holds that*

$$\texttt{Dec}_{sk}(\texttt{Enc}_{pk}(m; r)) = m\,.$$

[9] For simplicity here we only consider decryption with *implicit rejection*, that is, the output is a random value whenever the input is not a well-formed ciphertext for the particular sk. The extension to *explicit rejection* decryption can be done for example by adding a *flag bit* that marks the output as $\perp$ whenever decryption fails.

Definition 2.2.9 ($(1-\alpha)$-Correct PKE [DNR04])**.** *A* $(1-\alpha)$-correct PKE scheme*, or* PKE with decryption error α*, is a PKE scheme* $\text{PKE} = (\texttt{KGen}, \texttt{Enc}, \texttt{Dec})$ *such that, for any* $m \in \mathcal{M}$

$$\Pr_{\substack{(pk,sk)\leftarrow\$\,\texttt{KGen}() \\ r\leftarrow\$\,\mathcal{R}}}[\texttt{Dec}_{sk}(\texttt{Enc}_{pk}(m;r)) \neq m] \leq \alpha\,.$$

Below we define IND-CPA security for PKE schemes. This notion asks an adversary to distinguish whether an encryption oracle encrypts the left or the right message queried by the adversary.

Definition 2.2.10 (IND-CPA Security)**.** *Let* $\text{PKE} = (\texttt{KGen}, \texttt{Enc}, \texttt{Dec})$ *be a public key encryption scheme and the game* INDCPA *be as defined in Figure 2.10. For any adversary* $\mathcal{A}$*, its corresponding* IND-CPA *advantage is given by*

$$\mathbf{Adv}^{\mathsf{INDCPA}}_{\text{PKE}}(\mathcal{A}) \coloneqq \left|2\,\Pr[\mathsf{INDCPA}^{\mathcal{A}} \Rightarrow 1] - 1\right|\,.$$

Game INDCPA	**oracle** LR-Enc(m_0, m_1)
$b \leftarrow\$\,\{0,1\}$	$c \leftarrow\$\,\texttt{Enc}(pk, m_b)$
$(pk, sk) \leftarrow\$\,\texttt{KGen}()$	**return** c
$b' \leftarrow \mathcal{A}^{\mathsf{LR\text{-}Enc}}(pk)$	
return $(b' = b)$	

Figure 2.10: Security game INDCPA.

2.2.5 Digital Signatures

Signature schemes achieve authenticity, similar to MACs, the difference is that only the owner of the secret key can create a signature while anyone holding the public key can verify a signature. Below we formally define signature schemes.

Definition 2.2.11 (Signature Scheme)**.** *A signature scheme* $\Sigma = (\texttt{KGen}, \texttt{Sign}, \texttt{Verify})$ *is a tuple of three efficient algorithms such that:*

- $\texttt{KGen}\colon \mathbb{N} \to \mathcal{PK} \times \mathcal{SK}$ *is the key generation algorithm which takes a security parameter* λ *as input, and returns a keypair consisting of a public key* pk *and a secret key* sk*.*
- $\texttt{Sign}\colon \mathcal{SK} \times \mathcal{M} \to \mathcal{S}$ *is the signing algorithm which takes a secret key* sk *and a message* m*, and returns a signature* σ*.*
- $\texttt{Verify}\colon \mathcal{PK} \times \mathcal{M} \times \mathcal{S} \to \{0,1\}$ *is the verification algorithm which takes as input a public key* pk*, a message* m*, and a signature* σ*, and returns a bit.*

We denote the public key space, secret key space, message space, and signature space by $\mathcal{PK}$*,* $\mathcal{SK}$*,* $\mathcal{M}$*, and* $\mathcal{S}$*, respectively.*

A signature scheme is considered correct if honestly generated signatures are accepted by the verification algorithm. We formally define this below, accounting for some negligible error.

Definition 2.2.12. *A signature scheme* $\Sigma = (\texttt{KGen}, \texttt{Sign}, \texttt{Verify})$ *is correct if for any* $(pk, sk) \leftarrow_{\$} \texttt{KGen}(1^\lambda)$ *and any* $m \in \mathcal{M}$*, it holds that*

$$\Pr[\texttt{Verify}(pk, m, \texttt{Sign}(sk, m)) = 1] \geq 1 - \mathrm{negl}(\lambda)\,.$$

The standard security notion for signature schemes is *existential unforgeability under chosen message attacks* (EUF-CMA) which we define below.

Definition 2.2.13 (EUF-CMA Security)**.** *Let* $\Sigma = (\texttt{KGen}, \texttt{Sign}, \texttt{Verify})$ *be a signature scheme and game* EUFCMA *be as defined in Figure 2.11. For any adversary* $\mathcal{A}$ *its corresponding* EUF-CMA *advantage is given by*

$$\mathbf{Adv}^{\mathsf{EUFCMA}}_{\Sigma}(\mathcal{A}) \coloneqq \Pr[\mathsf{EUFCMA}^{\mathcal{A}} \Rightarrow 1]\,.$$

We say that Σ *is* EUF-CMA *secure if for any efficient adversary* $\mathcal{A}$ *it holds that*

$$\mathbf{Adv}^{\mathsf{EUFCMA}}_{\Sigma}(\mathcal{A}) \leq \mathrm{negl}(\lambda)\,.$$

Game EUFCMA	**oracle** $\mathsf{Sign}(m)$
$(pk, sk) \leftarrow_{\$} \texttt{KGen}(1^\lambda)$	$\mathcal{S} \leftarrow_{\cup} m$
$\mathcal{S} \leftarrow \emptyset$	$\sigma \leftarrow_{\$} \texttt{Sign}(sk, m)$
$(m, \sigma) \leftarrow \mathcal{A}^{\mathsf{Sign}}(pk)$	**return** σ
if $m \in \mathcal{S}$	
return 0	
return $\texttt{Verify}(pk, m, \sigma)$	

Figure 2.11: Security game EUFCMA.

We will use the notion of signature schemes with rerandomizable keys [FKM+16]. In the following we recall its definition.

Definition 2.2.14. *A signature scheme with perfectly rerandomizable keys* $\texttt{RSig}$ *is a tuple of algorithms* $(\texttt{RSig.KGen}, \texttt{RSig.Sign}, \texttt{RSig.Verify}, \texttt{RSig.RandPK}, \texttt{RSig.RandSK})$ *such that algorithms* $\texttt{RSig.KGen}$*,* $\texttt{RSig.Sign}$*, and* $\texttt{RSig.Verify}$ *satisfy the definition of a signature scheme (cf. Definition 2.2.11) and:*

- $\texttt{RSig.RandPK}\colon \mathcal{PK} \times \mathcal{R} \to \mathcal{PK}$ *is the public key rerandomization algorithm that takes as input the public key* pk *and a randomness* $r \in \mathcal{R}$ *and outputs a randomized public key* pk'*.*
- $\texttt{RSig.RandSK}\colon \mathcal{SK} \times \mathcal{R} \to \mathcal{SK}$ *is the secret key rerandomization algorithm that takes as input the secret key* sk *and a randomness* $r \in \mathcal{R}$ *and outputs a randomized secret key* sk'*.*

By $\mathcal{R}$ we denote the randomness space used to randomise the keys. Furthermore $\mathtt{RSig}$ *satisfies the following property. For all* $(pk, sk) \leftarrow\!\!\$\ \mathtt{KGen}(1^\lambda)$ *and for all* $r \in \mathcal{R}$ *it holds that* (pk_1, sk_1) *and* (pk_0, sk_0) *are identically distributed where* $pk_1 \leftarrow \mathtt{RSig.RandPK}(pk, r)$*,* $sk_1 \leftarrow \mathtt{RSig.RandSK}(sk, r)$*, and* $(pk_0, sk_0) \leftarrow\!\!\$\ \mathtt{RSig.KGen}(1^\lambda)$*.*

Correctness of a signature scheme with rerandomizable keys is defined analogously to standard signature schemes. The difference is that matching pairs of rerandomized public/secret keys have to satisfy correctness according to a standard signature scheme.

Definition 2.2.15. *A signature scheme with perfectly rerandomizable keys* $\mathtt{RSig}$ *is correct if for any* $(pk, sk) \leftarrow\!\!\$\ \mathtt{KGen}(1^\lambda)$*, any* $r \in \mathcal{R}$*, any* $m \in \mathcal{M}$*, it holds that:*

$$\Pr[\mathtt{RSig.Verify}(pk, m, \mathtt{RSig.Sign}(sk, m)) = 1] \geq 1 - \mathrm{negl}(\lambda)$$

and

$$\Pr[\mathtt{RSig.Verify}(pk_r, m, \mathtt{RSig.Sign}(sk_r, m)) = 1] \geq 1 - \mathrm{negl}(\lambda)\,,$$

where $pk_r \leftarrow \mathtt{RSig.RandPK}(pk, r)$ *and* $sk_r \leftarrow \mathtt{RSig.RandSK}(sk, r)$*.*

We also consider a relaxed version of Definition 2.2.14. Below we introduce the notion of signature schemes under rerandomizable public keys, where the property of rerandomizability of keys holds only for the generated public keys, but not in case of secret keys.

Definition 2.2.16. *A signature scheme with perfectly rerandomizable keys* $\mathtt{RSig}$ *consists of five algorithms* $(\mathtt{RSig.KGen}, \mathtt{RSig.Sign}, \mathtt{RSig.Verify}, \mathtt{RSig.RandPK}, \mathtt{RSig.RandSK})$ *such that algorithms* $\mathtt{RSig.KGen}$*,* $\mathtt{RSig.Sign}$*, and* $\mathtt{RSig.Verify}$ *satisfy the definition of a signature scheme (cf. Definition 2.2.11) and*

- $\mathtt{RSig.RandPK}$*:* $\mathcal{PK} \times \mathcal{R} \to \mathcal{PK}$ *is the public key rerandomization algorithm that takes as input the public key* pk *and a randomness* $r \in \mathcal{R}$ *and outputs a randomized public key* pk'*.*
- $\mathtt{RSig.RandSK}$*:* $\mathcal{SK} \times \mathcal{R} \to \mathcal{SK}$ *is the secret key rerandomization algorithm that takes as input the secret key* sk *and a randomness* $r \in \mathcal{R}$ *and outputs a randomized secret key* sk'*.*

By $\mathcal{R}$ we denote the randomness space. Furthermore $\mathtt{RSig}$ *satisfies the following property. For all* $(pk, sk) \leftarrow\!\!\$\ \mathtt{KGen}(1^\lambda)$ *and for all* $r \in \mathcal{R}$ *it holds that* pk_1 *and* pk_0 *are identically distributed where* $pk_1 \leftarrow \mathtt{RSig.RandPK}(pk, r)$ *and* $(pk_0, sk_0) \leftarrow\!\!\$\ \mathtt{RSig.KGen}(1^\lambda)$*.*

Below we define *existential unforgeability under chosen message attacks and honestly rerandomized keys* (EUF-CMA-HRK). The notion adapts the EUF-CMA security notion to signature schemes with rerandomizable keys.

Definition 2.2.17 (EUF-CMA-HRK Security)**.** *Let* $\mathtt{RSig}$ *be a signature scheme with perfectly rerandomizable keys and game* $\mathsf{EUFCMAHRK}$ *be as defined in Figure 2.12. For any adversary* $\mathcal{A}$ *its corresponding* EUF-CMA-HRK *advantage is given by*

$$\mathbf{Adv}^{\mathsf{EUFCMAHRK}}_{\mathtt{RSig}}(\mathcal{A}) := \Pr[\mathsf{EUFCMAHRK}^{\mathcal{A}} \Rightarrow 1]\,.$$

We say that $\mathtt{RSig}$ *is* EUF-CMA-HRK *secure if for any efficient adversary* $\mathcal{A}$ *it holds that*

$$\mathbf{Adv}^{\mathsf{EUFCMAHRK}}_{\mathtt{RSig}}(\mathcal{A}) \leq \mathrm{negl}(\lambda)\,.$$

Game EUFCMAHRK	**oracle** Rand()	**oracle** Sign(m, r)
$(pk, sk) \leftarrow\!\!\$\ \mathtt{RSig.KGen}(1^\lambda)$	$r \leftarrow\!\!\$\ \mathcal{R}$	**if** $r \notin \mathcal{T}$
$\mathcal{S} \leftarrow \emptyset$	$\mathcal{T} \leftarrow\!\!\cup\ r$	**return** $\bot$
$\mathcal{T} \leftarrow \emptyset$	**return** r	$\mathcal{S} \leftarrow\!\!\cup\ m$
$(m, \sigma, r) \leftarrow \mathcal{A}^{\mathsf{Sign},\mathsf{Rand}}(pk)$		$sk_r \leftarrow \mathtt{RSig.RandSK}(sk, r)$
if $m \in \mathcal{S}$		$\sigma \leftarrow\!\!\$\ \mathtt{RSig.Sign}(sk_r, m)$
return 0		**return** σ
if $r \notin \mathcal{T}$		
return 0		
$pk_r \leftarrow \mathtt{RSig.RandPK}(pk, r)$		
return $\mathtt{RSig.Verify}(pk, m, \sigma)$		

Figure 2.12: Security game EUFCMAHRK.

2.2.6 Pseudorandom Functions/Generators, and Hash Functions

In the following we recall pseudorandom functions, pseudorandom generators, and hash functions.

Below we define pseudorandomness for a keyed function $\mathcal{F}$. It asks an adversary to distinguish the function (keyed with a randomly chosen key K) from randomly chosen bit strings.

Definition 2.2.18 (PRF Security)**.** *Let $\mathcal{F}\colon \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ be a function family over the domain $\mathcal{X}$ and indexed by $\mathcal{K}$, and the game* PRF *be as defined in Figure 2.13. For any adversary $\mathcal{A}$ that never repeats a query to the oracle* F*, its corresponding* PRF *advantage is given by*

$$\mathbf{Adv}^{\mathsf{PRF}}_{\mathcal{F}}(\mathcal{A}) \coloneqq 2 \left| \Pr[\mathsf{PRF}^{\mathcal{A}} \Rightarrow 1] - 1 \right| .$$

Game PRF	**oracle** F(X)
$b \leftarrow\!\!\$\ \{0,1\}$	**if** $b = 0$
$K \leftarrow\!\!\$\ \mathcal{K}$	**return** $Y \leftarrow\!\!\$\ \mathcal{Y}$
$b' \leftarrow \mathcal{A}^{\mathsf{F}}()$	**else**
return $(b' = b)$	**return** $\mathcal{F}(K, X)$

Figure 2.13: Security game PRF.

Rather than sampling the random bit string on-the-fly, one can define an equivalent notion by initially picking a random function mapping from $\mathcal{X}$ to $\mathcal{Y}$ and granting the adversary oracle access. In this case, one can simply allow the adversary to repeat queries, although this is clearly pointless for the adversary. For the formalisation given above the game could maintain a list of queried points by the adversary to also allow repeating queries, in which case it would simply respond with the same response as the first time the query was made.

A pseudorandom generator is used to expand a short random bit string—typically called the seed—into a longer pseudorandom bit string. Note that our syntax defines a pseudorandom generator with variable output length, where the output length (in bits) is specified as part of the input. In addition, in our security definition we allow the adversary to make multiple queries to the challenge oracle G; by restricting the adversary to a single query, this covers the standard notion where the adversary only gets a single challenge.

Definition 2.2.19 (Pseudorandom Generators)**.** *Let $\mathcal{G}\colon \mathcal{S} \times \mathbb{N} \to \{0,1\}^*$ be a pseudorandom generator with an associated seed space $\mathcal{S}$, and let the game* PRG *be as defined in Figure 2.14. Then for any adversary $\mathcal{A}$, its corresponding* PRG *advantage is given by*

$$\mathbf{Adv}^{\mathsf{PRG}}_{\mathcal{G}}(\mathcal{A}) \coloneqq \left|2\Pr[\mathsf{PRG}^{\mathcal{A}} \Rightarrow 1] - 1\right| .$$

Game PRG	**oracle** $\mathsf{G}(L)$
$b \leftarrow\!\!\$\ \{0,1\}$	**if** $b = 0$
$b' \leftarrow \mathcal{A}^{\mathsf{G}}()$	$\quad Z \leftarrow\!\!\$\ \{0,1\}^L$
return $(b' = b)$	**else**
	$\quad z \leftarrow\!\!\$\ \mathcal{S}$
	$\quad Z \leftarrow \mathcal{G}(z, L)$
	return Z

Figure 2.14: Security game PRG.

Hash functions map arbitrary long inputs to outputs of fixed length. The common security notion is collision resistance which requires that it is hard to find a pair of colliding inputs. We define collision-resistant hash functions over a generic domain $\mathcal{X}$. Letting $\mathcal{X} = \{0,1\}^*$ results in the usual syntax but we can also, for instance, model a vector hash function over a triplet of strings by setting $\mathcal{X} = \{0,1\}^* \times \{0,1\}^* \times \{0,1\}^*$, which we will need in Chapter 3. For simplicity we only consider the random transformation model, where the adversary gets access to a random transformation ρ.

Definition 2.2.20 (Collision-Resistant Hash Functions)**.** *Let $\mathcal{H}\colon \mathcal{X} \to \{0,1\}^w$ be a hash function constructed from a random transformation ρ, with domain $\mathcal{X}$ and output length w. Then for any adversary $\mathcal{A}$ with oracle access to ρ, its corresponding advantage is given by*

$$\mathbf{Adv}^{\mathsf{CR}}_{\mathcal{H}}(\mathcal{A}) \coloneqq \Pr[\mathcal{H}(X_0) = \mathcal{H}(X_1) \wedge X_0 \neq X_1 \wedge X_0, X_1 \in \mathcal{X} \mid (X_0, X_1) \leftarrow \mathcal{A}^{\rho}()] .$$

2.2.7 Sponges

Sponges were initially introduced by Bertoni et al. [BDPVA07] as a tool to construct hash functions and it underpins the construction of SHA-3. However, sponges turned out to be a versatile tool that allows not only the construction of hash functions but also other primitives and several works provide primitives based on sponges [BDPV08, GPT15, DEM+17, DEM+20b, JLM14, BBB+20, DEMS16]. Besides the plain sponge

construction, Bertoni et al. [BDPV12] also introduce the duplex mode which is for instance used in [DMA17, MRV15].

At the core of the sponge construction is either a permutation π or a transformation ρ. In case of the former, it is said to be a P-sponge. Likewise, in case of the latter it is said to be a T-sponge. In this thesis we only consider T-sponges.

The sponge construction uses a state of n bits which constantly evolves. The state is divided into an outer state of r bits (called the rate) and an inner state of c bits (called the capacity) such that $r + c = n$. In its simplest form, the sponge construction initialises the state with the all-zero string. In the absorbing phase the input is "absorbed" by the sponge by XORing it r-bit at a time to the outer state followed by applying ρ. This procedure is repeated until the whole input is absorbed upon which the squeezing phase starts. During this phase the r-bit outer state is outputted followed by applying ρ until enough output bits are generated. This procedure is illustrated in Figure 2.15, using four rounds of absorbing and two rounds of squeezing.

The rate affects the efficiency of the sponge, a higher rate requires less rounds of absorbing/squeezing, while the capacity affects the security, a higher capacity yields a larger state that is unknown to an adversary.

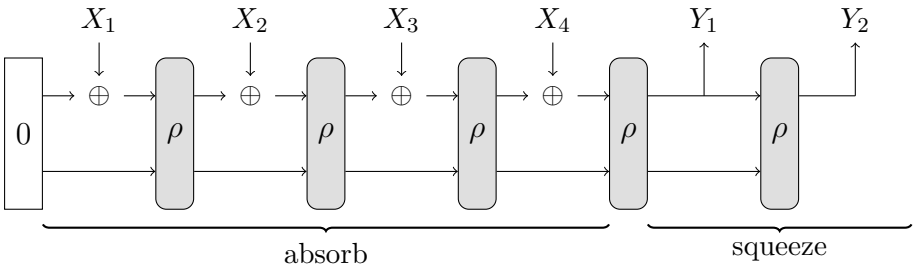

Figure 2.15: Plain sponge construction with four rounds of absorbing and two rounds of squeezing.

2.3 Leakage-Resilient Cryptography

We provide a description of the leakage model we are using in Chapter 3, existing security notions incorporating leakage, and existing results.

2.3.1 Leakage Model

Our leakage model is based on leakage resilience as defined in [DP08]. This follows the assumption by Micali and Reyzin [MR04] that only computation leaks, and in particular, that only the data that is accessed during computation can leak information. It allows for continuous adaptive leakage, where in each query to a leakage oracle the adversary can specify a leakage function from some predefined set $\mathcal{L}$ that it can choose adaptively based on prior outputs and leakage. Throughout, we restrict ourselves to leakage functions that are deterministic and efficiently computable. While our security definitions are formulated

in this general setting, our main results will be in the weaker *granular non-adaptive* leakage setting proposed in [FPS12]. We view the non-adaptive leakage setting as the special case where the leakage set $\mathcal{L}$ is restricted to be a singleton, fixed at the start of the game. In granular leakage, a single time step is with respect to a single computation of some underlying primitive, in our case, the transformation ρ. Correspondingly, in this case the adversary specifies a vector of leakage functions and gets in return the aggregated leakage from the entire evaluation of the higher-level construction. Note that in the granular setting the leakage sets for each iteration can be distinct. Similarly, when studying the leakage resilience of composite constructions we have to consider compositions of leakage functions. For instance, if construction C is composed of primitives A and B with associated leakage sets $\mathcal{L}_A$ and $\mathcal{L}_B$, then we associate to C the Cartesian product of the two leakage sets, i.e., $\mathcal{L}_C = \mathcal{L}_A \times \mathcal{L}_B$. The actual inputs that get fed to the leakage functions are implicitly defined by the construction and its inputs, whereas the combined output is the aggregate output of all function evaluations.

An analysis of sponge-based constructions compels us to consider leakage resilience in the random transformation model. A similar setting, albeit in the random oracle model, was already considered by Standaert et al. [SPY+10]. A central question that arises in idealised settings like this is whether the leakage function should be given access to the ideal primitive. As in [SPY+10], we will not give this access to the leakage function. On the one hand, providing the leakage function with unlimited access to the random oracle gives rise to artificial attacks, such as the "future computation attack" discussed in [SPY+10], that would not arise in practice. On the other hand, depriving the leakage function from accessing the ideal primitive, means that the leakage function cannot leak any bits of the ideal primitive's output, which may seem overly restrictive. However, for the case of sponge-based constructions this is less problematic because from the adversary's perspective the full output of a transformation call is completely determined by the input to the next transformation call. As such, information about the output of one transformation call can leak as part of the leakage in the next transformation call. Combined with the fact that the only restriction that we will impose on the leakage function is to limit its output length, we think that this leads to a fairly realistic leakage model.

We conclude our discussion on the leakage model by offering our interpretation of the significance of leakage resilience security with respect to practical side-channel attacks. One might object that we model leakage by a deterministic function whose output is of a fixed bit-length whereas in practice the leakage is noisy. However through the leakage function we are really trying to capture the maximum amount of information that an adversary may obtain from evaluating the scheme on a single input. Hence, the underlying assumption is that no matter how many times the scheme is run on the same input, in order to even out the noise, the information that the adversary can obtain is limited.

2.3.2 Security Notions incorporating Leakage

In the following we recall security notions which incorporate leakage as put forth by Barwell et al. [BMOS17].

Leakage-Resilient Function Families

Standard PRF security asks an adversary to distinguish whether its oracle implements the real function or just returns random bit strings. The corresponding leakage security notion is similar defined in that the challenge oracle stays the same. In addition, the adversary receives a learning oracle which always implements the real function and leaks. Repeating queries across the challenge and the learning oracle is forbidden as it results in a trivial attack.

Definition 2.3.1 (LPRF Security)**.** *Let $\mathcal{F}\colon \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ be a function family over the domain $\mathcal{X}$ and indexed by $\mathcal{K}$, and the game* LPRF *be as defined in Figure 2.16. For any adversary $\mathcal{A}$ that never forwards or repeats queries to or from the oracle* F *and only queries leakage functions in the set $\mathcal{L}_F$, its corresponding* LPRF *advantage is given by*

$$\mathbf{Adv}^{\mathsf{LPRF}}_{\mathcal{F}}(\mathcal{A}, \mathcal{L}_F) \coloneqq \left|2 \Pr[\mathsf{LPRF}^{\mathcal{A}} \Rightarrow 1] - 1\right| .$$

Game LPRF	**oracle** $\mathsf{F}(X)$	**oracle** $\mathsf{LF}(X, L)$
$b \leftarrow\!\!\$\ \{0,1\}$	**if** $b = 0$	$\Lambda \leftarrow L(K, X)$
$K \leftarrow\!\!\$\ \mathcal{K}$	**return** $Y \leftarrow\!\!\$\ \mathcal{Y}$	$Y \leftarrow \mathcal{F}(K, X)$
$b' \leftarrow \mathcal{A}^{\mathsf{F},\mathsf{LF}}()$	**else**	**return** (Y, Λ)
return $(b' = b)$	**return** $\mathcal{F}(K, X)$	

Figure 2.16: Security game LPRF.

Note that the notion can also be defined by picking a random function beforehand rather than generating it on-the-fly.

Chosen-Plaintext Security with Leakage

Ciphertext indistinguishability under chosen plaintext with leakage attacks (IND-CPLA) extends the classical IND-CPA security notion to the leakage setting. The adversary gets access to two oracles: (1) a challenge oracle which either implements the real encryption algorithm or returns random ciphertexts and (2) a learning oracle which implements the real encryption algorithm that additionally leaks.

Definition 2.3.2 (IND-CPLA Security)**.** *Let* $\mathrm{SE} = (\mathtt{Enc}, \mathtt{Dec})$ *be a symmetric encryption scheme and the game* INDCPLA *be as defined in Figure 2.17. For any adversary $\mathcal{A}$ that never forwards or repeats queries to or from the oracle* Enc *and only makes encryption queries containing leakage functions in the set $\mathcal{L}_E$, its corresponding* IND-CPLA *advantage is given by*

$$\mathbf{Adv}^{\mathsf{INDCPLA}}_{\mathrm{SE}}(\mathcal{A}, \mathcal{L}_E) \coloneqq \left|2 \Pr[\mathsf{INDCPLA}^{\mathcal{A}} \Rightarrow 1] - 1\right| .$$

Barwell et al. introduce a stronger variant, called *ciphertext indistinguishability under augmented chosen plaintext with leakage attacks* (IND-aCPLA) which grants the adversary an additional learning oracle. This additional oracle implements the real decryption oracle and also leaks. Access, however, is heavily restricted as the adversary can only query it on ciphertexts it previously received from its other learning oracle.

Game INDCPLA	**oracle** Enc(N, M)	**oracle** LEnc(N, M, L)
$b \leftarrow_\$ \{0,1\}$	$C \leftarrow \mathtt{Enc}(K, N, M)$	$\Lambda \leftarrow L(K, N, M)$
$K \leftarrow_\$ \mathcal{K}$	**if** $b = 0$	$C \leftarrow \mathtt{Enc}(K, N, M)$
$b' \leftarrow \mathcal{A}^{\mathsf{Enc},\mathsf{LEnc}}()$	**return** $C' \leftarrow_\$ \{0,1\}^{\lvert C\rvert}$	**return** (C, Λ)
return $(b' = b)$	**else**	
	return C	

Figure 2.17: Security game INDCPLA.

Definition 2.3.3 (IND-aCPLA Security)**.** *Let* $\mathrm{SE} = (\mathtt{Enc}, \mathtt{Dec})$ *be a symmetric encryption scheme and the game* INDaCPLA *be as defined in Figure 2.18. For any adversary* $\mathcal{A}$*, that never forwards or repeats queries to or from the oracle* Enc *and only makes leakage encryption and leakage decryption queries to its oracles* LEnc *and* LDec *containing leakage functions in the set* $\mathcal{L}_E$ *and* $\mathcal{L}_D$*, respectively, its corresponding* IND-aCPLA *advantage is given by*

$$\mathbf{Adv}^{\mathsf{INDaCPLA}}_{\mathrm{SE}}(\mathcal{A}, \mathcal{L}_E, \mathcal{L}_D) := \left|2 \Pr[\mathsf{INDaCPLA}^{\mathcal{A}} \Rightarrow 1] - 1\right| .$$

Game INDaCPLA	**oracle** Enc(N, M)	**oracle** LEnc(N, M, L)
$b \leftarrow_\$ \{0,1\}$	$C \leftarrow \mathtt{Enc}(K, N, M)$	$\Lambda \leftarrow L(K, N, M)$
$K \leftarrow_\$ \mathcal{K}$	**if** $b = 0$	$C \leftarrow \mathtt{Enc}(K, N, M)$
$b' \leftarrow \mathcal{A}^{\mathsf{Enc},\mathsf{LEnc},\mathsf{LDec}}()$	**return** $C' \leftarrow_\$ \{0,1\}^{\lvert C\rvert}$	**return** (C, Λ)
return $(b' = b)$	**else**	
	return C	**oracle** LDec(N, C, L)
		$\Lambda \leftarrow L(K, N, C)$
		$C \leftarrow \mathtt{Dec}(K, N, C)$
		return (C, Λ)

Figure 2.18: Security game INDaCPLA.

Note that both of the above notions are equivalent to IND-CPA security when dropping the leakage, i.e., considering an empty leakage set.

Unforgeability in the Presence of Leakage

To lift security of MACs to the leakage setting, Barwell et al. [BMOS17] introduce the notion of *strong unforgeability under chosen message with leakage attacks* (SUF-CMLA). It grants the adversary a leakage-free challenge oracle which either implements the real verification algorithm or always rejects. Additionally, the adversary gets two learning oracles which leak. One implements the tagging algorithm while the other implements the verification algorithm. The adversary is not allowed to forward any output from the leaking tagging oracle. It is, however, permitted to make a query to the leakage verification oracle and then repeat it to the challenge oracle.

Definition 2.3.4 (SUF-CMLA Security)**.** *Let* MAC = (Tag, Ver) *be a message authentication code and the game* SUFCMLA *be as defined in Figure 2.19. For any adversary* $\mathcal{A}$ *that never forwards queries to or from the oracle* Vfy*, and only queries leakage functions to its oracles* LTag *and* LVfy *in the respective sets* $\mathcal{L}_T$ *and* $\mathcal{L}_V$*, its corresponding* SUF-CMLA *advantage is given by*

$$\mathbf{Adv}^{\mathsf{SUFCMLA}}_{\mathrm{MAC}}(\mathcal{A}, \mathcal{L}_T, \mathcal{L}_V) \coloneqq \left|2\ \Pr[\mathsf{SUFCMLA}^{\mathcal{A}} \Rightarrow 1] - 1\right| .$$

Game SUFCMLA	**oracle** Vfy(X, T)	**oracle** LTag(X, L)
$b \leftarrow\!\!\$\ \{0,1\}$	**if** $b = 0$	$\Lambda \leftarrow L(K, X)$
$K \leftarrow\!\!\$\ \mathcal{K}$	**return** $\perp$	$T \leftarrow \mathtt{Tag}(K, X)$
$b' \leftarrow \mathcal{A}^{\mathsf{Vfy},\mathsf{LTag},\mathsf{LVfy}}()$	**else**	**return** (T, Λ)
return $(b' = b)$	$V \leftarrow \mathtt{Ver}(K, X, T)$	
	return V	**oracle** LVfy(X, T, L)
		$\Lambda \leftarrow L(K, X, T)$
		$V \leftarrow \mathtt{Ver}(K, X, T)$
		return (V, Λ)

Figure 2.19: Security game SUFCMLA.

Authenticated Encryption with Leakage

Classical security for AEAD schemes is lifted to the leakage setting yielding LAE security. The adversary gets access to four oracles. Two leakage-free challenge oracles which are the same as in standard AE security; they implement either the real encryption and decryption or return random bit strings and always reject ciphertexts. Furthermore, the adversary gets access to two learning oracles which leak. These oracles implement the real encryption and decryption. Trivial attacks are excluded by disallowing any repeating/forwarding of queries to or from the two challenge oracles.

Definition 2.3.5 (LAE Security)**.** *Let* AEAD = (Enc, Dec) *be an authenticated encryption scheme with associated data and the game* LAE *be as defined in Figure 2.20. For any nonce-respecting adversary* $\mathcal{A}$ *that never forwards or repeats queries to or from the oracles* Enc *and* Dec *and only makes encryption and decryption queries containing leakage functions in the respective sets* $\mathcal{L}_{AE}$ *and* $\mathcal{L}_{VD}$*, describing the leakage sets for authenticated encryption and verified decryption, its corresponding* LAE *advantage is given by*

$$\mathbf{Adv}^{\mathsf{LAE}}_{\mathrm{AEAD}}(\mathcal{A}, \mathcal{L}_{AE}, \mathcal{L}_{VD}) \coloneqq \left|2\ \Pr[\mathsf{LAE}^{\mathcal{A}} \Rightarrow 1] - 1\right| .$$

The following theorem states the LAE security of the N2 schemes. It shows that it reduces to the security of the underlying encryption scheme and MAC. Precisely, the IND-aCPLA security of the encryption scheme, the SUF-CMLA security of the MAC, and the LPRF security of the tagging algorithm.

Game LAE

$b \leftarrow_{\$} \{0,1\}$
$K \leftarrow_{\$} \mathcal{K}$
$b' \leftarrow \mathcal{A}^{\mathsf{Enc},\mathsf{LEnc},\mathsf{Dec},\mathsf{LDec}}()$
return $(b' = b)$

oracle $\mathsf{Enc}(N, A, M)$

$C \leftarrow \mathtt{Enc}(K, N, A, M)$
if $b = 0$
  return $C' \leftarrow_{\$} \{0,1\}^{|C|}$
else
  return C

oracle $\mathsf{LEnc}(N, A, M, L)$

$\Lambda \leftarrow L(K, N, A, M)$
$C \leftarrow \mathtt{Enc}(K, N, A, M)$
return (C, Λ)

oracle $\mathsf{Dec}(N, A, C)$

if $b = 0$
  return $\perp$
$M \leftarrow \mathtt{Dec}(K, N, A, C)$
return M

oracle $\mathsf{LDec}(N, A, C, L)$

$\Lambda \leftarrow L(K, N, A, C)$
$M \leftarrow \mathtt{Dec}(K, N, A, C)$
return (M, Λ)

Figure 2.20: Security game LAE.

Theorem 2.3.6 (LAE Security of the N2 Construction [BMOS17, Theorem 1])**.** *Let* $\textsc{Se} = (\mathtt{Enc}, \mathtt{Dec})$ *be a symmetric encryption scheme with associated leakage sets* $(\mathcal{L}_E, \mathcal{L}_D)$ *and* $\textsc{Mac} = (\mathtt{Tag}, \mathtt{Ver})$ *be a MAC with associated leakage sets* $(\mathcal{L}_T, \mathcal{L}_V)$*. Further let* N2 *be the composition of* Se *and* Mac *described in Figure 2.7, with associated leakage sets* $(\mathcal{L}_{AE}, \mathcal{L}_{VD})$ *where* $\mathcal{L}_{AE} = \mathcal{L}_E \times \mathcal{L}_T$ *and* $\mathcal{L}_{VD} = \mathcal{L}_D \times \mathcal{L}_V$*. Then for any* LAE *adversary* $\mathcal{A}_{\mathsf{ae}}$ *against* N2 *there exist adversaries* $\mathcal{A}_{\mathsf{se}}$*,* $\mathcal{A}_{\mathsf{prf}}$*, and* $\mathcal{A}_{\mathsf{mac}}$ *such that:*

$$\mathbf{Adv}^{\mathsf{LAE}}_{\mathrm{N2}}(\mathcal{A}_{\mathsf{ae}}, \mathcal{L}_{AE}, \mathcal{L}_{VD}) \leq \mathbf{Adv}^{\mathsf{INDaCPLA}}_{\textsc{Se}}(\mathcal{A}_{\mathsf{se}}, \mathcal{L}_E, \mathcal{L}_D) + \mathbf{Adv}^{\mathsf{LPRF}}_{\mathsf{Tag}}(\mathcal{A}_{\mathsf{prf}}, \mathcal{L}_T) + 2\,\mathbf{Adv}^{\mathsf{SUFCMLA}}_{\textsc{Mac}}(\mathcal{A}_{\mathsf{mac}}, \mathcal{L}_T, \mathcal{L}_V)\,.$$

Note that the other two schemes (N1 and N3) are not generally secure in the leakage setting [BMOS17]. The problem is that both authenticate the message whereas N2 authenticates the ciphertext. This allows N2 to verify ciphertexts without using the decryption of the underlying encryption scheme. Both N1 and N3 always have to run the decryption algorithm to verify the ciphertext—and reject if verification fails. Since this decryption algorithm potentially leaks, the adversary can collect leakage even from invalid ciphertexts. This thwarts a composition theorem following paradigms different than Encrypt-then-MAC.

2.3.3 Min-Entropy

Below we recall a definition and a result on min-entropy that we will need in the proofs of Theorem 3.2.1 and Theorem 3.2.2.

Definition 2.3.7. *Let* X*,* Y*, and* Z *be random variables. The average-case guessing probability of* X *conditioned on* Y *and* $Z = z$ *is given by*

$$GP(X \mid Y, Z = z) \coloneqq \mathbb{E}_{y \leftarrow Y}\left[\max_x \Pr[X = x \mid Y = y \cap Z = z]\right]\,.$$

The average-case min-entropy of X *conditioned on* Y *and* $Z = z$ *is defined to be:*

$$\mathrm{H}_\infty\left(X \mid Y, Z = z\right) \coloneqq -\log(GP(X \mid Y, Z = z))\,.$$

Lemma 2.3.8 ([DORS08])**.** *Let X, Y, and Z be random variables such that the support of Y is bounded above by 2^λ. Then*

$$\mathrm{H}_\infty(X \mid Y, Z = z) \geq \mathrm{H}_\infty(X, Y \mid Z = z) - \lambda \geq \mathrm{H}_\infty(X \mid Z = z) - \lambda\,.$$

2.4 Related-Key Attacks

We recall some of the existing related-key attacks (RKA) security notions. All notions follow the style introduced by Bellare and Kohno [BK03]. That is, the set of admissible related-key-deriving (RKD) functions is fixed at the start of the game. All our results, however, also apply to the alternative definition given by Harris [Har11], where the adversary first picks the set of RKD functions before the concrete scheme (from a family of primitives) is chosen by the game. This prevents an inherent limitation of the Bellare-Kohno formalism as the RKD function can not depend on the primitive.[10]

Φ-restricted Adversaries. For RKA security notions, the adversary is typically restricted to a set of functions that it can query to its oracles. This restriction is necessary, as Bellare and Kohno [BK03] showed that RKA security is unachievable without such restrictions. Let $\mathcal{K}$ be the key space of some primitive, then the set of permitted RKD functions is $\Phi \subset \mathsf{Func}(\mathcal{K}, \mathcal{K})$. We call an adversary, that only queries functions from the set Φ to its oracles, a *Φ-restricted adversary*. Note, in particular, that the case of Φ only containing the identity function restores the standard setting of a single key.

Repeating Queries. To avoid trivial wins, certain queries must be excluded from the security games. In case of a MAC, we must forbid the adversary to query its challenge verification oracle on a tag it obtained from its tagging oracle. To do this, the security game can keep a list of such queries and let the verification oracle check for such forbidden queries; in which case it will simply reject the query. Another option, is to simply exclude adversaries that perform such queries in the security definition. For ease of exposition, we follow the latter approach.

RKA Security for MACs and Pseudorandom Functions. We give the definition of related-key attack security of MACs. Existing notions define it as an unforgeability game where the adversary finally outputs a forgery attempt [BR14, Xag13]. We define unforgeability of a MAC against RKA as a distinguishing game. Here the adversary aims to distinguish whether its challenge verification oracle implements the real verification algorithm or simply rejects any queried tag.

Definition 2.4.1 (SUF-CMRKA Security)**.** *Let $\Gamma = (\mathtt{Tag}, \mathtt{Ver})$ be a message authentication code and $\Phi \subset \mathsf{Func}(\mathcal{K}, \mathcal{K})$. Let the game* SUFCMRKA *be defined as in Figure 2.21. For a Φ-restricted RKA adversary $\mathcal{A}$, that never forwards a query from its oracle* Tag*, we define its* SUF-CMRKA *advantage as*

$$\mathbf{Adv}^{\mathsf{SUFCMRKA}}_{\Gamma}(\mathcal{A}, \Phi) \coloneqq \left|2\,\Pr[\mathsf{SUFCMRKA}^{\mathcal{A}} \Rightarrow 1] - 1\right|\,.$$

Game SUFCMRKA	**oracle** $\mathsf{Vfy}(X, T, \phi)$	**oracle** $\mathsf{Tag}(X, \phi)$
$b \leftarrow_\$ \{0,1\}$	**if** $b = 0$	$T \leftarrow \mathtt{Tag}(\phi(K), X)$
$K \leftarrow_\$ \mathcal{K}$	**return** $\perp$	**return** T
$b' \leftarrow \mathcal{A}^{\mathsf{Vfy},\mathsf{Tag}}()$	**else**	
return $(b' = b)$	**return** $\mathtt{Ver}(\phi(K), X, T)$	

Figure 2.21: Security game SUFCMRKA.

RKA security for pseudorandom functions (PRFs) has been studied in many works, e.g., [BK03, BC10, BCM11, AFPW11], and are defined as the advantage in distinguishing the real function from a random function when having access to an oracle implementing either of these enhanced with RKD functions.

Definition 2.4.2 (RKA-PRF Security)**.** *Let $\mathcal{F}\colon \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ and $\Phi \subset \mathsf{Func}(\mathcal{K}, \mathcal{K})$. Let the game* rkaPRF *be defined as in Figure 2.22. For a Φ-restricted RKA adversary $\mathcal{A}$, that never repeats a query, we define its* RKA-PRF *advantage as*

$$\mathbf{Adv}^{\mathsf{rkaPRF}}_{\mathcal{F}}(\mathcal{A}, \Phi) := \left| 2 \Pr[\mathsf{rkaPRF}^{\mathcal{A}} \Rightarrow 1] - 1 \right| .$$

Games rkaPRF	$\mathsf{F}(X, \phi)$ in rkaPRF
$b \leftarrow_\$ \{0,1\}$	**if** $b = 0$
$K \leftarrow_\$ \mathcal{K}$	$Y \leftarrow_\$ \mathcal{F}^*(\phi(K), X)$
$\mathcal{F}^* \leftarrow_\$ \mathsf{Func}(\mathcal{K}, \mathcal{X}, \mathcal{Y})$	**else**
$b' \leftarrow \mathcal{A}^{\mathsf{F}}()$	$Y \leftarrow \mathcal{F}(\phi(K), X)$
return $(b' = b)$	**return** Y

Figure 2.22: Security games rkaPRF.

2.5 Related-Randomness Attacks

Yilek [Yil10b] defines security against resetting attacks, which extends the standard notion of IND-CPA.[11] This models a scenario in which the randomness used to encrypt might be reused, for instance, when performed on a virtual machine using snapshots. The security game is displayed in Figure 2.23. The adversary gets access to a challenge left-or-right oracle LR-Enc and aims to distinguish which of its messages it encrypts. In addition, the adversary gets an encryption oracle Enc, which allows to encrypt using arbitrary, adversarial chosen public keys. The crucial part is that the adversary specifies an index for both oracles to determine which randomness is used to encrypt, i.e., repeating an index

[10] It is questionable whether RKD functions that depend on the actual primitive are relevant in practice.
[11] In [Yil10b] the notion is also extended to the IND-CCA case, which is not relevant for this thesis.

results in a repeated randomness. This is also the reason for the additional encryption oracle. In the classical IND-CPA setting, there is no need for this oracle since the adversary can encrypt locally.

Game $\mathsf{lrINDCPRA}$	**oracle** $\mathsf{LR\text{-}Enc}(m_0, m_1, i)$	**oracle** $\mathsf{Enc}(pk, m, i)$
$b \leftarrow_\$ \{0,1\}$	**if** $f[i] = \bot$	**if** $f[i] = \bot$
$(sk^*, pk^*) \leftarrow_\$ \mathtt{KGen}()$	$\quad f[i] \leftarrow_\$ \mathcal{R}$	$\quad f[i] \leftarrow_\$ \mathcal{R}$
$b' \leftarrow \mathcal{A}^{\mathsf{LR\text{-}Enc},\mathsf{Enc}}(pk^*)$	$r^* \leftarrow f[i]$	$r^* \leftarrow f[i]$
return $(b' = b)$	**if** $b = 0$	$c \leftarrow \mathtt{Enc}(pk, m; r^*)$
	$\quad c \leftarrow \mathtt{Enc}(pk^*, m_0; r^*)$	**return** c
	else	
	$\quad c \leftarrow \mathtt{Enc}(pk^*, m_1; r^*)$	
	return c	

Figure 2.23: Security game $\mathsf{lrINDCPRA}$ using different randomnesses. Note that we will never use this game due to Lemma 2.5.1 which shows that it can be simplified to a single randomness.

Yilek [Yil10b] gives two lemmas to simplify the notion. One lemma shows that we can restrict the notion to a single randomness which is used for every encryption query by the adversary. The other lemma, identified as flawed in [PSS14], claims that we can restrict the adversary to a single query to its left-or-right oracle $\mathsf{LR\text{-}Enc}$. Below we recall the former.

Lemma 2.5.1 ([Yil10b, Lemma 1])**.** *Let* $\textsc{Pke}$ *be a PKE scheme and the game* $\mathsf{lrINDCPRA}$ *be defined as in Figure 2.23. Then for any adversary* $\mathcal{A}$*, making* q *queries to* $\mathsf{LR\text{-}Enc}$ *and querying in total* t *different indices to* $\mathsf{LR\text{-}Enc}$ *and* Enc*, there exists an adversary* $\mathcal{B}$*, making* q *queries to* $\mathsf{LR\text{-}Enc}$ *and querying only a single index to* $\mathsf{LR\text{-}Enc}$ *and* Enc *such that*

$$\mathbf{Adv}^{\mathsf{lrINDCPRA}}_{\textsc{Pke}}(\mathcal{A}) \leq t\, \mathbf{Adv}^{\mathsf{lrINDCPRA}}_{\textsc{Pke}}(\mathcal{B})\,.$$

Lemma 2.5.1 allows to simplify the security game by choosing one randomness r^* at the beginning of the game, which is used for every query by the adversary. The simplified security game $\mathsf{lrINDCPRA}$ is displayed in Figure 2.24.

To exclude trivial wins, Yilek [Yil10b] defines equality-pattern-respecting adversaries. Intuitively, these are adversaries which never repeat a message to their encryption oracles and do not make two challenge queries which are equal in the left message but different in the right message, or vice versa. Below we formally define such adversaries.[12] Note that this definition is necessary to achieve a meaningful security notion, but is not an immoderate restriction imposed on the adversary.

Definition 2.5.2. *Let* $\mathcal{A}$ *be an adversary playing* $\mathsf{lrINDCPRA}$ *which makes* q *queries to* $\mathsf{LR\text{-}Enc}$*. Let* $\mathcal{E}$ *be the set of messages* m *such that* $\mathcal{A}$ *makes a query* (pk^*, m) *to* Enc*.*

[12] Note that this definition is tailored to the single randomness setting. The equivalent, more complicated definition for multiple randomnesses (see, e.g., [Yil10b, Appendix A]) is irrelevant, as we focus on the single randomness setting, and therefore omitted.

Let $(m_0^1, m_1^1), \ldots, (m_0^q, m_1^q)$ *be the queries to* LR-Enc. *We say that* $\mathcal{A}$ *is* equality-pattern-respecting *if*

- *for all* $b \in \{0,1\}$ *and* $i \in [q]$, $m_b^i \notin \mathcal{E}$ *and*
- *for all* $b \in \{0,1\}$ *and* $i \neq j$, $m_b^i = m_b^j \implies m_{1-b}^i = m_{1-b}^j$.

The LoR-IND-CPRA advantage of an adversary is defined as follows.

Definition 2.5.3. *Let* $\textsc{Pke} = (\mathtt{KGen}, \mathtt{Enc}, \mathtt{Dec})$ *be a public key encryption scheme and the game* lrINDCPRA *be defined as in Figure 2.24. For any equality-pattern-respecting adversary* $\mathcal{A}$*, its* LoR-IND-CPRA *advantage is defined as*

$$\mathbf{Adv}_{\textsc{Pke}}^{\mathsf{lrINDCPRA}}(\mathcal{A}) \coloneqq \left|2 \Pr[\mathsf{lrINDCPRA}^{\mathcal{A}} \Rightarrow 1] - 1\right| .$$

Game lrINDCPRA	**oracle** LR-Enc(m_0, m_1)	**oracle** Enc(pk, M)
$b \leftarrow\!\!\$\ \{0,1\}$	**if** $b = 0$	$c \leftarrow \mathtt{Enc}(pk, m; r^*)$
$(sk^*, pk^*) \leftarrow\!\!\$\ \mathtt{KGen}()$	$\quad c \leftarrow \mathtt{Enc}(pk^*, m_0; r^*)$	**return** c
$r^* \leftarrow\!\!\$\ \mathcal{R}$	**else**	
$b' \leftarrow \mathcal{A}^{\mathsf{LR\text{-}Enc},\mathsf{Enc}}(pk^*)$	$\quad c \leftarrow \mathtt{Enc}(pk^*, m_1; r^*)$	
return $(b' = b)$	**return** c	

Figure 2.24: Security game lrINDCPRA.

2.6 Quantum-Resistant Cryptography

In this section we provide the necessary background for Part II. We first give basics on quantum computation, followed by introducing the quantum random oracle model (QROM). We then recall existing quantum security notions as well as some quantum-resistant cryptographic primitives.

2.6.1 Quantum Computation

A quantum system, identified by a letter A, is represented by a complex Hilbert space, which we denote by $\mathfrak{H}_A$. If A is clear from the context, we drop the subscript and write $\mathfrak{H}$ rather than $\mathfrak{H}_A$. Pure states in a Hilbert space $\mathfrak{H}$ are representatives of equivalence classes of elements of $\mathfrak{H}$ of norm 1. Mixed states, on the other hand, are a more general representation of quantum states that takes entanglement with external systems into account; they are elements of the density matrix operator space over $\mathfrak{H}$, that is, Hermitian positive semi-definite linear operators of trace 1, denoted as $\mathfrak{D}(\mathfrak{H})$. We use the bra-ket notation (Dirac notation [Dir39]) for pure states, e.g., $|\phi\rangle$, while mixed states will be denoted by lowercase Greek letters, e.g., ρ. Operations on pure states from A to B are performed by applying a unitary operator $U\colon \mathfrak{H}_A \to \mathfrak{H}_B$ to the state, while the more general case of operations on mixed states is described by superoperators of the form $U\colon \mathfrak{D}(\mathfrak{H}_A) \to \mathfrak{D}(\mathfrak{H}_B)$.

The canonical way to compute a classical function $\mathcal{F}\colon \mathcal{X} \to \mathcal{Y}$ on a superposition of possible inputs $\sum_{x\in\mathcal{X}} \alpha_x |x\rangle$ is through the so-called type-1 operator for $\mathcal{F}$ described by:

$$U_{\mathcal{F}}^{(1)}\colon \sum_{x,y} \alpha_{x,y} |x,y\rangle \mapsto \sum_{x,y} \alpha_{x,y} |x, y \oplus \mathcal{F}(x)\rangle \ .$$

This can always be implemented efficiently whenever $\mathcal{F}$ is efficient [NC16]. By linearity, it is sufficient to specify just the behaviour on the basis elements, i.e.:

$$U_{\mathcal{F}}^{(1)}\colon |x,y\rangle \mapsto |x, y \oplus \mathcal{F}(x)\rangle \ .$$

If $\mathcal{F}$ is invertible, then there is another, non-equivalent way to compute $\mathcal{F}$ in superposition. This is done through the so-called type-2 operators, which are defined as the unitary:

$$U_{\mathcal{F}}^{(2)}\colon |x\rangle \mapsto |\mathcal{F}(x)\rangle \ .$$

See Figure 2.25 for an illustration of these different operators. Kashefi et al. [KKVB02] first introduced type-2 operators using the term minimal oracles instead. They show that these operators are strictly stronger by giving a problem, which can be solved exponentially faster with type-2 operators than with type-1 operators. They also observe that the adjoint of the type-2 operator corresponds to the type-2 operator of the inverse function $\mathcal{F}^{-1}$, which is (usually) not the case for type-1 operators. Besides that, type-2 operators have been used by Gagliardoni et al. [GHS16] to define quantum security for symmetric encryption schemes.

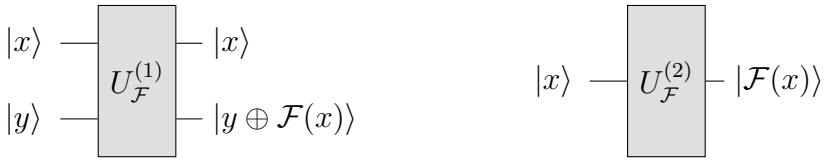

Figure 2.25: **Left:** type-1 operator for $\mathcal{F}$. **Right:** type-2 operator for $\mathcal{F}$.

An important result of quantum mechanics is the no-cloning theorem [WZ82, Die82]. It states that there is no physical process, which upon receiving an arbitrary quantum state $|\phi\rangle$, outputs two copies of said state.[13] The no-cloning theorem poses some challenges when considering quantum adversaries as queries cannot simply be copied by a reduction. A typical example where the no-cloning theorem has a positive effect is quantum money [Zha19b, Zha21, AC12, Wie83]. Here, banknotes are quantum states and counterfeit banknotes are thwarted by the no-cloning theorem.

2.6.2 Quantum Random Oracle Model

The *random oracle model* (ROM), formalized by Bellare and Rogaway [BR93], is a commonly used model to prove cryptographic schemes secure. In the ROM, all parties have

[13] Note that copying of orthogonal states such as $|0\rangle/|1\rangle$ or $|+\rangle/|-\rangle$ is possible as they can be perfectly distinguished by measuring in the correct basis upon which one can prepare two copies of the measured state.

access to a random oracle H which, upon being queried on a value x, returns a random value y. Every further query of x, for instance by another party, is answered using the same y as before. When a scheme is proven secure in the ROM, one idealises components like hash functions by a random oracle. Given that the code of a hash function is publicly available, one has to assume that a quantum adversary implements hash functions on its quantum computer, thereby being able to evaluate it in superposition. This assumption gives rise to the *quantum random oracle model* (QROM), which has been advocated by Boneh et al. [BDF+11]. In the QROM, parties which have quantum computing power are allowed to query the random oracle in superposition. In more detail, for a random oracle H, the QROM allows these parties access to the quantum random oracle $|\mathsf{H}\rangle$, where $|\mathsf{H}\rangle : |x, y\rangle \mapsto |x, y \oplus \mathsf{H}(x)\rangle$. To prove a scheme post-quantum secure, the proof should always be in the QROM, as a proof in the ROM would imply the unrealistic expectation that the adversary refrains from implementing a hash function on its quantum computer. We use superscripts to denote oracle access, e.g., $\mathcal{A}^{\mathsf{H}}$ and $\mathcal{A}^{|\mathsf{H}\rangle}$ for the ROM and QROM, respectively.

In our proofs we also consider reprogrammed random oracles. For a random oracle H, we denote the random oracle which is reprogrammed on input x to y by $\mathsf{H}_{x \mapsto y}$, i.e.,

$$\mathsf{H}_{x \mapsto y}(a) = \begin{cases} y & \text{, if } a = x \\ \mathsf{H}(a) & \text{, else} \end{cases} .$$

An important tool for the QROM is the *one-way to hiding* (O2H) lemma developed by Unruh [Unr15c]. It relates the distinguishing advantage between two random oracles, which differ only in some subset of inputs, to the probability of querying an input from said subset. Below we recall the O2H lemma, albeit in the formulation put forth by Ambainis et al. [AHU19].

Lemma 2.6.1 (One-way to hiding (O2H) [AHU19])**.** *Let G, $\mathsf{H} \colon \mathcal{X} \to \mathcal{Y}$ be random functions, let z be a random bit string, and let $\mathcal{S} \subset \mathcal{X}$ be a random set such that $\forall x \notin \mathcal{S}$, $\mathsf{G}(x) = \mathsf{H}(x)$. $(\mathsf{G}, \mathsf{H}, \mathcal{S}, z)$ may have arbitrary joint distribution. Furthermore, let $\mathcal{A}^{\mathsf{H}}$ be a quantum oracle algorithm which queries H at most q times. Define an oracle algorithm $\mathcal{B}^{\mathsf{H}}$ as follows: Pick $i \leftarrow\!\!\$\, [q]$. Run $\mathcal{A}^{\mathsf{H}}_{\mathsf{q}}(z)$ until just before its i^{th} query to H. Measure the query in the computational basis, and output the measurement outcome. Let*

$$\begin{aligned} P_{left} &:= \Pr[\mathcal{A}^{\mathsf{H}}(z) \Rightarrow 1] \\ P_{right} &:= \Pr[\mathcal{A}^{\mathsf{G}}(z) \Rightarrow 1] \\ P_{guess} &:= \Pr[x \in \mathcal{S} \,|\, \mathcal{B}^{\mathsf{H}}(z) \Rightarrow x] . \end{aligned}$$

Then it holds that

$$|P_{left} - P_{right}| \leq 2q\sqrt{P_{guess}} .$$

The same result holds with $\mathcal{B}^{\mathsf{G}}(z)$ instead of $\mathcal{B}^{\mathsf{H}}(z)$ in the definition of P_{guess}.

Following the original O2H lemma by Unruh, several works have introduced variants of the lemma. These variants achieve better bounds, yet at the cost of a more restricted applicability [AHU19, BHH+19, KSS+20].

The importance of the O2H lemma and its different variants can be seen by a large series of works, that use it in order to prove post-quantum security of several different cryptographic primitives [BL20, MSS21, CMP20, DFMS21, Mar20, SRP18, SY17, TU16, XAY+20, Zha20, CRSS20, CCL20, CCLY21, CMSZ19, HKSU20, HXY19, JZM19a, JZM19b, KKPP20, LW21].

Another tool that we will use are Zhandry's small range distributions, defined below. These are distributions where the set of possible outputs is limited.

Definition 2.6.2 (Small-range distributions [Zha12a])**.** *Let $\mathcal{X}$, $\mathcal{Y}$ be sets, D be a distribution on $\mathcal{Y}$, r be an integer, P be a random function from $\mathcal{X}$ to $[r]$, and $\mathbf{y} = (y_1, \ldots, y_r)$ be r samples of D. Define a function $\mathsf{H}\colon \mathcal{X} \to \mathcal{Y}$ by $x \mapsto y_{P(x)}$. The distribution of H, induced by P and $\mathbf{y}$, is called a* small-range distribution *with r samples of D.*

The following lemma provides a bound on the distinguishing advantage between a random oracle and an oracle drawn from a small-range distribution when superposition access is granted.

Lemma 2.6.3 ([Zha12a])**.** *There is a universal constant C such that, for any set $\mathcal{X}$ and $\mathcal{Y}$, distribution D on $\mathcal{Y}$, integer l, and any quantum algorithm $\mathcal{A}$ making q queries to an oracle $\mathsf{H}\colon \mathcal{X} \to \mathcal{Y}$, the following two cases are indistinguishable, except with probability less than $\frac{Cq^3}{l}$:*

- *$\mathsf{H}(x) = y_x$ where $\mathbf{y}$ is a list of samples of D of size $|\mathcal{X}|$.*
- *H is drawn from the small-range distribution with l samples of D.*

2.6.3 Security Notions

We recall the security notion *ciphertext indistinguishability under quantum chosen plaintext attacks* (IND-qCPA) as given by Boneh and Zhandry [BZ13b]. This notion grants the adversary full superposition access in the learning phase while restricting the challenge phase to be classical. Note that this notion is equivalent to post-quantum security for public key encryption schemes as there is no oracle in the learning phase.[14] Below we define it for symmetric key encryption schemes. Note that the exposition is changed to make the randomness explicit, which is important for the results in Chapter 6.

Definition 2.6.4 (IND-qCPA Security)**.** *Let $\mathrm{SE} = (\mathtt{Enc}, \mathtt{Dec})$ be a symmetric encryption scheme and the game $\mathsf{INDqCPA}$ be as defined in Figure 2.26. For any adversary $\mathcal{A}$, its corresponding* IND-qCPA *advantage is defined as*

$$\mathbf{Adv}^{\mathsf{INDqCPA}}_{\mathrm{SE}}(\mathcal{A}) \coloneqq \left|2\Pr[\mathsf{INDqCPA}^{\mathcal{A}} \Rightarrow 1] - 1\right| .$$

Gagliardoni et al. [GHS16] extended security to the case where the challenge messages can be in superposition. To circumvent an impossibility result by Boneh and Zhandry [BZ13b], they make use of type-2 operators which directly work on the message register rather than on a separate output register. A core part of [GHS16] is to show

[14] We emphasise that Boneh and Zhandry also introduce a variant which grants superposition access to the decryption oracle which is not equivalent to post-quantum security.

Game INDqCPA	**oracle** LR-Enc(m_0, m_1)	**oracle** Enc$(\lvert\phi\rangle = \sum_{m,c} \alpha_{m,c} \lvert m\rangle \lvert c\rangle)$
$b \leftarrow\!\!\$\ \{0,1\}$	$r \leftarrow\!\!\$\ \mathcal{R}$	$r \leftarrow\!\!\$\ \mathcal{R}$
$k \leftarrow\!\!\$\ \mathcal{K}$	$c \leftarrow \mathtt{Enc}(k, m_b; r)$	$\sum_{m,c} \lvert r\rangle \lvert m\rangle \lvert c \oplus \mathtt{Enc}_k(m; r)\rangle \leftarrow U^{(1)}_{\mathtt{Enc}}(\lvert r\rangle \lvert\phi\rangle)$
$b' \leftarrow \mathcal{A}^{\mathsf{LR\text{-}Enc},\mathsf{Enc}}()$	**return** c	trace out $\lvert r\rangle$
return $(b' = b)$		**return** $\sum_{m,c} \lvert m\rangle \lvert c \oplus \mathtt{Enc}_k(m; r)\rangle$

Figure 2.26: Security game INDqCPA.

that these operators can be realised efficiently. Below we give the simplest form of their security notion, that is, *quantum ciphertext indistinguishability* (qIND), which allows a single challenge query and no learning queries.

Definition 2.6.5 (qIND Security)**.** *Let* $\mathrm{SE} = (\mathtt{Enc}, \mathtt{Dec})$ *be a symmetric encryption scheme and the game* qIND *be as defined in Figure 2.27. For any adversary* $\mathcal{A}$*, making exactly one query to its oracle* LR-Enc*, its corresponding* qIND *advantage is defined as*

$$\mathbf{Adv}^{\mathsf{qIND}}_{\mathrm{SE}}(\mathcal{A}) \coloneqq \left|2 \Pr[\mathsf{qIND}^{\mathcal{A}} \Rightarrow 1] - 1\right| .$$

Game qIND	**oracle** LR-Enc$(\lvert\phi_0\rangle, \lvert\phi_1\rangle)$, where $\lvert\phi_i\rangle = \sum_m \alpha_{m,i} \lvert m\rangle$
$b \leftarrow\!\!\$\ \{0,1\}$	$r \leftarrow\!\!\$\ \mathcal{R}$
$k \leftarrow\!\!\$\ \mathcal{K}$	$\lvert r\rangle \lvert c\rangle \leftarrow U^{(2)}_{\mathtt{Enc}}(\lvert r\rangle \lvert\phi_b\rangle \lvert 0^{\gamma-\mu}\rangle)$
$b' \leftarrow \mathcal{A}^{\mathsf{LR\text{-}Enc}}()$	trace out $\lvert r\rangle$
return $(b' = b)$	**return** $\lvert c\rangle$

Figure 2.27: Security game qIND.

2.6.4 Quantum-Resistant Public Key Encryption

We recall several public key encryption schemes based on quantum hard assumptions as well as the generic hybrid encryption scheme

Lattice-Based Hardness Assumptions

We recall the hardness assumption on which the lattice-based schemes considered in this thesis are based. Below we start with the *learning with errors* (LWE) problem.

Problem 2.6.6 (Learning With Errors)**.** *Let* n, m, q *be integers,* χ *be some distribution over* $\mathbb{Z}$*, and* $\mathsf{s}, \mathsf{e} \leftarrow \chi$*. Given* (a, b)*, where* $\mathsf{a} \leftarrow\!\!\$\ \mathbb{Z}_q^n$ *and* $\mathsf{b} \leftarrow \mathsf{a}\mathsf{s} + \mathsf{e}$*, find* s*.*

The exact hardness of the problem depends on the distribution χ, in general, the secrets and the errors are small.

There is also a decisional variant of the LWE problem, called *decisional learning with errors* (DLWE). Here, one is given either an LWE sample or a random sample and asked to distinguish these cases.

Problem 2.6.7 (Decisional Learning With Errors)**.** *Let n, m, q be integers and χ be some distribution over $\mathbb{Z}$ and $\mathsf{s}, \mathsf{e} \leftarrow \chi$. Given (a, b), where $\mathsf{a} \leftarrow_{\$} \mathbb{Z}_q^n$ and either $\mathsf{b} \leftarrow \mathsf{as} + \mathsf{e}$ or $\mathsf{b} \leftarrow_{\$} \mathbb{Z}_q$, distinguish these cases.*

Additional variants of the LWE problem are module LWE [BGV12, LS15], middle product LWE [RSSS17], and LWE in the exponent [DGG16, Göp16].

LWE-Based Public Key Encryption Scheme

We recall the canonical LWE-based PKE scheme due to Regev [Reg05, Reg09]. It is displayed in Figure 2.28, for sake of simplicity we give the scheme in a generic form without specifying concrete sets. It underpins many lattice-based public key encryption schemes such as Kyber [SAB+20, BDK+18], FrodoKEM [NAB+19], LIMA [AOP+17, SAL+17], LAC [LLJ+19], the LP scheme [LP11], NewHope [PAA+19, ADPS16], Saber [DKR+20], and Round5 [GZB+19]. The security of the scheme is based on the DLWE problem (cf. Problem 2.6.7).

The scheme works on n-dimensional vectors of elements of $\mathbb{Z}_q$ for $q \geq 2$. The functions Encode and Decode are used for encoding and decoding bit strings to and from elements of $\mathbb{Z}_q^n$. The Decode function has a certain error tolerance which, upon being exceeded, results in a decryption failure.

The Encode function maps the bits of the message into the high-order bit representation of group elements, which are then represented as a vector. For our purpose, it is not important here to have a precise definition of this encoding function, nor to have a detailed discussion on the sampling distribution of the LWE vectors—which is generally crucial for proving the security of the scheme. We leave these details to an appropriate reference, for example [NAB+19]. We will only consider the following, simplified characterisation of the encoding function.

Lemma 2.6.8 (Canonical LWE-Based Message Encoding)**.** *Let $\mathsf{Encode}\colon m \mapsto \mathsf{v}$ as from Figure 2.28, and let $\mathsf{Bit}(\mathsf{v})$ be the canonical bit string representation in use for a vector element v over a finite group $\mathbb{Z}_q$. Then there exists a public efficient invertible permutation π and an integer $\tau \geq 0$ such that*

$$\mathsf{Bit}(\mathsf{Encode}(m)) = \pi\left(m \| \overbrace{0 \ldots 0}^{\tau}\right).$$

In particular, for $q = 2$, it holds $\tau = 0$ and $\mathsf{Bit}(\mathsf{Encode}(m)) = \pi(m)$.

Notice that in practice the parameter τ denotes the expansion factor between m and c_1 in Figure 2.28 and is upper bounded by $n \cdot \lceil \log_2(q) \rceil$ minus the bit size of the message. The larger τ, the less efficient the scheme is in terms of ciphertext size but the lower the decryption failure rate.

$\texttt{KGen}(\lambda; r)$	$\texttt{Enc}(pk, m; r)$
$\mathsf{a}, \mathsf{s}, \mathsf{e} \leftarrow r$	**parse** pk **as** (a, b)
$\mathsf{b} \leftarrow \mathsf{as} + \mathsf{e}$	$\mathsf{e}_1, \mathsf{e}_2, \mathsf{d} \leftarrow r$
$pk \leftarrow (\mathsf{a}, \mathsf{b})$	$c_1 \leftarrow \mathsf{bd} + \mathsf{e}_1 + \mathsf{Encode}(m)$
$sk \leftarrow \mathsf{s}$	$c_2 \leftarrow \mathsf{ad} + \mathsf{e}_2$
return (pk, sk)	**return** $c \leftarrow (c_1, c_2)$

Figure 2.28: Pseudocode of the canonical LWE-based public key encryption scheme PKE with algorithms $(\texttt{KGen}, \texttt{Enc}, \texttt{Dec})$. For the randomness r used by $\texttt{KGen}$ and $\texttt{Enc}$, let $x \leftarrow r$ denote that x is deterministically derived from r. The decryption algorithm is omitted, as it is irrelevant for this thesis.

$\texttt{KGen}()$	$\texttt{Enc}^{\mathsf{H}}(pk, m = (m_1, m_2, m_3))$
$\mathsf{a}_1, \mathsf{a}_2 \leftarrow\!\!\$\ \mathcal{R}_q$	$\mathbf{c} \leftarrow\!\!\$\ \mathbb{Z}_p^n$
$\mathsf{r}_1, \mathsf{r}_2 \leftarrow \mathcal{D}_{\mathbb{Z}^n, r_{sec}}$	$\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3, \mathbf{d} \leftarrow \mathsf{H}(\mathbf{c})$
$\mathsf{a}_3 \leftarrow p^{k-1} - (\mathsf{a}_1\mathsf{r}_1 + \mathsf{a}_2\mathsf{r}_2)$	$\mathbf{s} \leftarrow \mathbf{c} + p\mathbf{d}$
$sk \leftarrow (\mathsf{r}_1, \mathsf{r}_2)$	$\mathbf{t}_i \leftarrow \mathsf{Encode}(m_i) + \mathbf{v}_i \bmod w$ **for** $i \in [3]$
$pk \leftarrow (\mathsf{a}_1, \mathsf{a}_2, \mathsf{a}_3)$	$\mathbf{e}_i \leftarrow \mathcal{D}_{\mathbf{t}_i + w\mathbb{Z}^n, s}$ **for** $i \in [3]$
return (pk, sk)	$\mathsf{b}_i \leftarrow \mathsf{a}_i\mathbf{s} + \mathbf{e}_i$ **for** $i \in [3]$
	$c \leftarrow (\mathsf{b}_1, \mathsf{b}_2, \mathsf{b}_3)$
	return c

Figure 2.29: Pseudocode of the lattice-based public key encryption scheme LARA. The decryption algorithm is omitted since it is irrelevant for this thesis.

Lattice-based Public Key Encryption Scheme LARA

Another lattice-based public key encryption scheme we consider is LARA [Ban19]. The scheme is displayed in Figure 2.29, where q is an integer and n is a power of 2. The polynomial ring $\mathbb{Z}_q[X]/\langle X^n + 1\rangle$ is denoted by $\mathcal{R}_q$. The security is based on a variant of the LWE problem (cf. Problem 2.6.6) and the DLWE problem (cf. Problem 2.6.7).

We refer to [Ban19] for the parameters s, w, p, and r_{sec}. LARA uses the discrete Gaussian distribution, which is denoted by $\mathcal{D}_{x,\sigma}$, where x and σ are the support and standard deviation, respectively. Multiplications are considered to be polynomial multiplications. Vectors and polynomials are transformed into one another by setting the coefficients to the vector entries and vice versa. The scheme uses an encoding function Encode, similar to the LWE-based PKE scheme described above, which maps messages to polynomials.

Code-Based Hardness Assumptions

We recall the code-based hardness assumption relevant for the code-based public key encryption schemes we consider. Note that we do not define codes as it is not relevant for this work. Note further that we only recall the hardness assumptions for ROLLO-II [ABD+19];

for the other two schemes, HQC [AAB+19a] and RQC [AAB+19b], we provide attacks in Chapter 4, which are independent of the underlying hardness assumptions.

In the *ideal rank support recovery* (IRSR) problem, one is given a vector $\mathbf{h}$, a polynomial P, and a syndrome $\boldsymbol{\sigma}$, and asked to find a support E containing vectors $\mathbf{e}_1$, $\mathbf{e}_2$ such that $\mathbf{e}_1 + \mathbf{e}_2\mathbf{h} = \boldsymbol{\sigma} \bmod \mathsf{P}$.

Problem 2.6.9 (2-Ideal Rank Support Recovery [ABD+19])**.** *Given a vector $\mathbf{h} \in \mathbb{F}_{q^{\overline{m}}}^n$, a polynomial $\mathsf{P} \in \mathbb{F}_q[X]$ of degree n and a syndrome $\boldsymbol{\sigma}$, and a weight w, find a support E of dimension lower than w such that $\mathbf{e}_0 + \mathbf{e}_1\mathbf{h} = \boldsymbol{\sigma} \bmod \mathsf{P}$ where the vectors are of support E.*

The *ideal low rank parity check* (ILRPC) codes indistinguishability problem asks to distinguish whether a vector $\mathbf{h}$ is sampled uniformly at random or computed as $\mathbf{x}^{-1}\mathbf{y} \bmod \mathsf{P}$, for vectors $\mathbf{x}$, $\mathbf{y}$ of small dimension.

Problem 2.6.10 (Ideal Low Rank Parity Check Codes Indistinguishability [ABD+19])**.** *Given a polynomial $\mathsf{P} \in \mathbb{F}_q[X]$ of degree n and a vector $\mathbf{h} \in \mathbb{F}_{q^{\overline{m}}}^n$, distinguish whether the ideal code $\mathcal{C}$ with the parity-check matrix generated by $\mathbf{h}$ and P is a random ideal code or if it is an ideal LRPC code of weight d.*

In other words, distinguish whether the vector $\mathbf{h}$ was sampled uniformly at random or as $\mathbf{x}^{-1}\mathbf{y} \bmod \mathsf{P}$ where vectors $\mathbf{x}$ and $\mathbf{y}$ have the same support of small dimension d.

Code-Based Public Key Encryption Scheme ROLLO-II

The encryption scheme ROLLO-II [ABD+19] is a code-based public key encryption scheme based on rank metric codes. The scheme in a generic, simplified form is displayed in Figure 2.30, where H is a random oracle, Supp describes the support of vectors, and P is a polynomial from the underlying code problem.

Throughout, p is a prime and q is some power of p. For an integer $\overline{m}$, the finite field that contains $q^{\overline{m}}$ elements is $\mathbb{F}_{q^{\overline{m}}}$ and the corresponding vector space of dimension n is given by $\mathbb{F}_{q^{\overline{m}}}^n$. The set of vectors of length n with rank weight w over the set $\mathbb{F}_{q^{\overline{m}}}$ is denoted by $\mathcal{S}_w^n(\mathbb{F}_{q^{\overline{m}}})$, where the rank weight of a vector is the rank of a specific matrix associated with that vector (see [ABD+19] for more details). Below we define the support of a word.

Definition 2.6.11. *Let $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{F}_{q^{\overline{m}}}^n$. The support E of $\mathbf{x}$, denoted $\mathsf{Supp}(\mathbf{x})$, is the $\mathbb{F}_q$-subspace of $\mathbb{F}_{q^{\overline{m}}}$ generated by the $\mathbf{x}$, i.e., $\mathsf{E} = \langle x_1, \ldots, x_n \rangle_{\mathbb{F}_q}$.*

Multiplications are considered to be polynomial multiplications, where vectors and polynomials are transformed into one another by taking the vector entries as coefficients and vice versa. In the scheme, d and r are integers while P is an irreducible polynomial over $\mathbb{F}_{q^{\overline{m}}}$.

Code-Based Public Key Encryption Scheme HQC

We describe the code-based PKE scheme HQC [AAB+19a]. The binary field is denoted by $\mathbb{F}_2$. For a vector $\mathbf{x}$, $\omega(\mathbf{x})$ defines its hamming weight, i.e., the number of coordinates different than 0. The scheme uses global parameters $(n, \overline{m}, \delta, w, w_{\mathbf{r}}, w_{\mathbf{e}})$ which depend on the security parameter. By $\mathcal{R}_2$ we denote $\mathbb{F}_2[X]/(X^n - 1)$. The scheme is displayed in Figure 2.31.

$\texttt{KGen}()$	$\texttt{Enc}^{\mathsf{H}}(pk, m)$
$\mathbf{x}, \mathbf{y} \leftarrow\!\!\$\ \mathcal{S}_d^{2n}(\mathbb{F}_{q^m})$	$\mathbf{e}_1, \mathbf{e}_2 \leftarrow\!\!\$\ \mathcal{S}_r^{2n}(\mathbb{F}_{q^m})$
$\mathbf{h} \leftarrow \mathbf{x}^{-1}\mathbf{y} \bmod \mathsf{P}$	$\mathsf{E} \leftarrow \mathsf{Supp}(\mathbf{e}_1, \mathbf{e}_2)$
$sk \leftarrow (\mathbf{x}, \mathbf{y})$	$c_1 \leftarrow m \oplus \mathsf{H}(\mathsf{E})$
$pk \leftarrow \mathbf{h}$	$c_2 \leftarrow \mathbf{e}_1 + \mathbf{e}_2\mathbf{h} \bmod \mathsf{P}$
return (pk, sk)	**return** $c \leftarrow (c_1, c_2)$

Figure 2.30: Pseudocode of the code-based public key encryption scheme ROLLO-II. The decryption algorithm is omitted since it is irrelevant for this thesis.

$\texttt{KGen}()$	$\texttt{Enc}(pk, m)$
$\mathbf{h} \leftarrow\!\!\$\ \mathcal{R}_2$	**parse** pk **as** $(\mathbf{h}, \mathbf{s}, \mathbf{G})$
$\mathbf{G} \leftarrow\!\!\$\ \mathbb{F}_2^{k \times n}$	$\mathbf{e} \leftarrow\!\!\$\ \mathcal{R}_2$ **s.t.** $\omega(\mathbf{e}) = w_{\mathbf{e}}$
$\mathbf{x}, \mathbf{y} \leftarrow\!\!\$\ \mathcal{R}_2^2$ **s.t.** $\omega(\mathbf{x}) = \omega(\mathbf{y}) = w$	$(\mathbf{r}_1, \mathbf{r}_2) \leftarrow\!\!\$\ \mathcal{R}_2^2$ **s.t.** $\omega(\mathbf{r}_1) = \omega(\mathbf{r}_2) = w_{\mathbf{r}}$
$\mathbf{s} \leftarrow \mathbf{x} + \mathbf{h}\mathbf{y}$	$c_1 \leftarrow m\mathbf{G} + \mathbf{s}\mathbf{r}_2 + \mathbf{e}$
$pk \leftarrow (\mathbf{h}, \mathbf{s}, \mathbf{G})$	$c_2 \leftarrow \mathbf{r}_1 + \mathbf{h}\mathbf{r}_2$
$sk \leftarrow (\mathbf{x}, \mathbf{y})$	**return** $c \leftarrow (c_1, c_2)$
return (pk, sk)	

Figure 2.31: Pseudocode of the code-based public key encryption scheme HQC. The decryption algorithm is omitted since it is irrelevant for this thesis.

Code-Based Public Key Encryption Scheme RQC

We describe the code-based public key encryption scheme RQC [AAB+19b]. It uses global parameters $(n, k, \delta, w, w_r, \mathsf{P})$, where P is an irreducible polynomial over $\mathbb{F}_q[X]$. All global parameters depend on the security parameter. By $\mathbb{F}_q$ and $\mathbb{F}_{q^{\overline{m}}}$, we denote the finite field containing q and $q^{\overline{m}}$ elements, respectively. The set of vectors of length n and rank weight w over $\mathbb{F}_{q^{\overline{m}}}$ is denoted by $\mathcal{S}_w^n(\mathbb{F}_{q^{\overline{m}}})$. Likewise, $\mathcal{S}_{1,w}^n(\mathbb{F}_{q^{\overline{m}}})$ describes the set of vectors of length n of rank weight w such that its support contains 1. The public key encryption scheme is displayed in Figure 2.32.

$\texttt{KGen}()$	$\texttt{Enc}(pk, m)$
$\mathbf{h} \leftarrow\!\!\$\ \mathbb{F}_{q^{\overline{m}}}$	**parse** pk **as** $(\mathbf{h}, \mathbf{s}, \mathbf{G})$
$\mathbf{G} \leftarrow\!\!\$\ \mathcal{S}_n^n(\mathbb{F}_{q^{\overline{m}}})$	$\mathbf{e}, \mathbf{r}_1, \mathbf{r}_2 \leftarrow\!\!\$\ \mathcal{S}_{w_r}^{3n}(\mathbb{F}_{q^{\overline{m}}})$
$(\mathbf{x}, \mathbf{y}) \leftarrow\!\!\$\ \mathcal{S}_{1,w}^{2n}(\mathbb{F}_{q^{\overline{m}}})$	$c_1 \leftarrow m\mathbf{G} + \mathbf{s}\mathbf{r}_2 + \mathbf{e}$
$\mathbf{s} \leftarrow \mathbf{x} + \mathbf{h}\mathbf{y}$	$c_2 \leftarrow \mathbf{r}_1 + \mathbf{h}\mathbf{r}_2$
$pk \leftarrow (\mathbf{h}, \mathbf{s}, \mathbf{G})$	**return** $c \leftarrow (c_1, c_2)$
$sk \leftarrow (\mathbf{x}, \mathbf{y})$	
return (pk, sk)	

Figure 2.32: Pseudocode of the code-based public key encryption scheme RQC. The decryption algorithm is omitted since it is irrelevant for this thesis.

Hybrid Encryption Scheme

The canonical hybrid encryption scheme combines a public key encryption and a symmetric key encryption scheme into a public key encryption scheme. That is, a message is encrypted using a fresh one-time key of the symmetric encryption scheme. The one-time key is then encrypted using the public key encryption scheme, whereupon the encrypted one-time key is attached to the ciphertext. To decrypt, one first recovers the symmetric one-time key, and then uses it to decrypt the ciphertext containing the message. The canonical hybrid encryption scheme is shown in Figure 2.33.

$\texttt{KGen}()$	$\texttt{Enc}(pk, m)$	$\texttt{Dec}(sk, c)$
$(pk, sk) \leftarrow\!\!\$\ \texttt{KGen}^P()$	$k \leftarrow\!\!\$\ \mathcal{K}$	**parse** c **as** (c_1, c_2)
return (pk, sk)	$c_1 \leftarrow\!\!\$\ \texttt{Enc}^S(k, m)$	$k \leftarrow \texttt{Dec}^P(sk, c_2)$
	$c_2 \leftarrow\!\!\$\ \texttt{Enc}^P(pk, k)$	$m \leftarrow \texttt{Dec}^S(k, c_1)$
	return (c_1, c_2)	**return** m

Figure 2.33: Hybrid encryption scheme $\textsc{Pke} = (\texttt{KGen}, \texttt{Enc}, \texttt{Dec})$ built from a PKE scheme $\Sigma^P = (\texttt{KGen}^P, \texttt{Enc}^P, \texttt{Dec}^P)$ and an SKE scheme $\Sigma^S = (\texttt{Enc}^S, \texttt{Dec}^S)$.

Part I

Security in Hostile Environments

CHAPTER 3

LEAKAGE-RESILIENT CRYPTOGRAPHY

In this chapter, we develop the FGHF′ *construction, a generic blueprint for leakage-resilient authenticated encryption schemes with associated data; and* SLAE*, a concrete instantiation of the former built entirely from sponges. The* FGHF′ *construction (cf. Section 3.1) and* SLAE *(cf. Section 3.2) are based on [DJS19], the improvements of the* FGHF′ *construction (cf. Section 3.3) are based on [KS20b].*

Contents

3.1 FGHF′**: A Generic Construction** **46**
3.2 SLAE**: An Instantiation based on Sponges** **58**
3.3 LAE **from** LPRF **84**
3.4 Summary and Outlook **100**

Cryptographic security proofs often consider the adversary to access a primitive in a black-box way, i.e., it has only access to the input/output behaviour but not the internal working. In practice, however, an adversary might be able to learn more than just the input/output behaviour. This extra information, often referred to as leakage, can be obtained via so-called side-channel analysis (SCA) which can stem from running time [Koc96], electromagnetic radiation [GMO01, QS01, RR01], power consumption [KJJ99], and fault detection [BS97, BDL97]. Such leakage can be devastating and lead to complete breaks of cryptographic primitives [EKM+08].

This sparked the research area of leakage-resilient cryptography [DP08]. Here, the leakage is modelled via a leakage function which is limited to a certain number of output bits. As a simple example one can think of a function that outputs the first bit of

the secret key. The leakage function models the information that the adversary can obtain from SCA, irrespectively of the exact type of SCA. Although there are many works regarding the leakage resilience of cryptographic primitives, they often consider block ciphers [Pie09, DP10, YSPY10, FPS12, MSJ12, SPY13, MSNF16, GSWY19, PSP+08], which only support messages of fixed length and do not provide authenticity of ciphertexts.

In this chapter, we extend the research of leakage-resilient cryptography to authenticated encryption schemes with associated data. As one of the most fundamental cryptographic primitives, authenticated encryption with associated data is a well-studied cryptographic primitive in a leakage-free setting. Over the last years, the study of AEAD schemes has received a lot of attention, for instance through the CAESAR competition [Ber14] or the ongoing NIST standardization process on lightweight cryptography [NIST15]. However, the leakage resilience of AEAD schemes is not that well understood, but recently, several AEAD schemes, which are designed to be secure in the presence of side-channel leakage, have been proposed [DEM+17, BMOS17, BPPS17, BGP+19, GPPS20, DM19].

A notable example is the work by Barwell et al. [BMOS17] which provides constructions coming with strong security guarantees. However, this comes at the cost of a complex construction and less efficiency. More precisely, they achieve security against adaptive leakage, but require several invocations of the underlying leakage-resilient pseudorandom function per encryption, which makes it rather costly.

A more practical solution is the authenticated encryption scheme Isap, proposed by Dobraunig et al. [DEM+17]. Entirely built from sponges, the design of Isap is very simple and thus striving for a more practical solution. It follows the conventional Encrypt-then-MAC approach [BN00, BN08], by relying on the N2 construction [NRS14], augmented by a rekeying strategy inspired by [MSGR10]. However, the design choices of Isap are predominantly heuristic, lacking any formal security analysis to justify its claims.

Contribution

In this chapter, we extend this line of research with the goal of providing leakage-resilient AEAD constructions that are both practical—like Isap—and are backed by a formal security analysis—like the constructions in [BMOS17].[15] We develop the FGHF′ construction, a generic blueprint for achieving leakage-resilient AEAD schemes from leakage-resilient function families. The FGHF′ construction refines the N2 construction by decomposing the symmetric encryption scheme and message authentication code into smaller building blocks. We further provide Slae, a leakage-resilient AEAD scheme constructed entirely from sponges. Admittedly, Slae bears many similarities with, and is indeed inspired by, Isap. However, Slae is conceptually different than Isap. In particular, Slae is understood through a top-down approach, where the leakage-resilient AEAD scheme is gradually decomposed into simpler building blocks that are finally instantiated using sponges. However, Slae does not rely on a rekeying strategy as is the case for Isap.

Both Slae and the generic FGHF′ construction are analysed in the leakage-resilient

[15] We would like to point out that, subsequent to our work, Isap v2.0 [DEM+20b] has been developed which comes with a security proof following the framework by Barwell et al. [BMOS17].

cryptography framework given by Dziembowski and Pietrzak [DP08], using the leakage-resilient security notions put forth by Barwell et al. [BMOS17].

Finally, we recast the N2 composition theorem by Barwell et al. (cf. Theorem 2.3.6) for a special type of encryption schemes and MACs which include the schemes underlying the FGHF′ construction. Hereby, we can relax the security requirements towards the underlying building blocks of the FGHF′ construction.

Related Work

Leakage-resilient cryptographic primitives, ranging from encryption schemes and message authentication codes to authenticated encryption schemes, have been proposed in several works [BKP+16, BPPS17, GPPS19, BGP+19, GPPS20, PSV15]. In contrast to our setting, these works allow leakage on the challenge queries. However, some of the underlying components are assumed to be leak-free, which is typically achieved using techniques such as masking [CJRR99, GP99, SP06]. A subset of these works also assume that the leakage is simulatable, an assumption that is not beyond dispute [SPY13, LMO+14]. Functions and permutations which are pseudorandom under leakage have been proposed for instance in [DP10, FPS12, SPY+10, YS13].

Abdalla, Belaïd, and Fouque [ABF13] construct a symmetric encryption scheme that is non-adaptively leakage-resilient against chosen-plaintext attacks. Interestingly, their scheme employs a rekeying function that is not a leakage-resilient pseudorandom function. However, their encryption scheme is not nonce-based as it necessitates a source of randomness.

Agrawal et al. developed RCB [ABC+18], which, to the best of our knowledge, claimed to be the first leakage-resilient authenticated encryption scheme. RCB, however, was broken shortly after by Abed et al. [ABL17] who showed that RCB does not provide authenticity, even in the leakage-free setting.

Independent and concurrent to this thesis, Dobraunig and Mennink [DM19] analyse the leakage resilience of the duplex sponge construction. They show that the duplex is indistinguishable from an *adjusted ideal extendible input function* (AIXIF), which is an ideal functionality incorporating leakage, which is different from our results, although their leakage model is close to ours.

Also, independent and concurrent to this thesis, Guo et al. [GPPS20] propose an AEAD design, TETSponge, that combines a sponge construction with two tweakable block cipher instances. While their work shares the same goal, i.e., constructing leakage-resilient AEAD schemes, they use a different approach. More specifically, they rely on the tweakable block cipher instances being leak-free, for instance via a hardened implementation, to achieve leakage resilience of the AEAD scheme.

Subsequent to the results in this chapter, Isap v2.0 has been developed [DEM+20b]. It comes with small tweaks to increase security against implementation attacks. More relevant, in comparison to this work, Isap v2.0 comes with a security proof following the framework by Barwell et al. [BMOS17], whereas the security of Isap was predominantly heuristic.

Dobrauning and Mennink [DM21] consider the question of leakage-resilient value comparison. This is particularly relevant for message authentication where validity of a tag is

done by recomputing the tag and then comparing it with the candidate tag. They show different means on how to do this in the context of leakage.

Roadmap

In Section 3.1 we develop the FGHF′ construction. The sponge-based instantiation SLAE is given and analysed in Section 3.2. Section 3.3 provides improvements of the FGHF′ construction while Section 3.4 concludes with a summary and outlook for further research.

3.1 FGHF′: A Generic Construction

In this section, we develop the FGHF′ construction, which is a generic blueprint for leakage-resilient authenticated encryption schemes with associated data. The FGHF′ construction is a refinement of the N2 construction (cf. Figure 2.7), which combines a nonce-based symmetric encryption scheme and a MAC into an authenticated encryption scheme with associated data. Barwell et al. [BMOS17] showed that security of the N2 construction extends to the leakage setting. More precisely, they showed that if the encryption component is IND-aCPLA secure and the vector MAC is both LPRF and SUF-CMLA secure, then the composition is LAE secure. In turn, the FGHF′ construction further breaks down the encryption component and the MAC component of N2, denoted by $\textsc{Se}[\mathcal{F}, \mathcal{G}]$ and $\textsc{Mac}[\mathcal{H}, \mathcal{F}']$, respectively, into smaller parts. Namely encryption is realised from a fixed-input-length leakage-resilient pseudorandom function $\mathcal{F}$ and a standard pseudorandom generator $\mathcal{G}$, whereas the MAC is realised from a vector hash function $\mathcal{H}$, and a fixed-input-length function $\mathcal{F}'$, that is both leakage-resilient pseudorandom and leakage-resilient unpredictable.

The FGHF′ construction in terms of the symmetric encryption scheme $\textsc{Se}[\mathcal{F}, \mathcal{G}]$ and message authentication code $\textsc{Mac}[\mathcal{H}, \mathcal{F}']$, which are in turn composed of fixed-input-length functions $\mathcal{F}$ and $\mathcal{F}'$, a pseudorandom generator $\mathcal{G}$ and a vector hash function $\mathcal{H}$, is depicted in Figure 3.1. The pseudocode is given in Figure 3.3.

The structure of FGHF′, being an instance of N2, allows to apply the N2 composition theorem by Barwell et al. [BMOS17] (cf. Theorem 2.3.6). In order to prove the FGHF′ construction LAE secure, we need to show that the underlying components meet the requirements of Theorem 2.3.6, i.e., $\textsc{Se}[\mathcal{F}, \mathcal{G}]$ is IND-aCPLA secure, $\textsc{Mac}[\mathcal{H}, \mathcal{F}']$ is SUF-CMLA secure, and $\textsc{Mac}[\mathcal{H}, \mathcal{F}']$, or rather its tagging algorithm $\mathtt{Tag}[\mathcal{H}, \mathcal{F}']$, is an LPRF.

As it turns out, we can realise an IND-aCPLA secure encryption directly from an LPRF and a variable-output-length PRG. Here the PRG serves only to extend the range of the LPRF in order for the encryption scheme to accommodate variable-length messages. Surprisingly, a standard PRG without any leakage resilience suffices.

Contrary to the leakage-free setting, where pseudorandomness and unpredictability are equivalent [NR98], the latter property is not automatically implied by the former when a MAC is constructed from an LPRF through the canonical construction. This is because the game SUFCMLA is more permissive than the game LPRF with respect to the adversary's queries. Namely, the adversary can forward queries from the oracle LVfy to oracle Vfy, whereas in the game LPRF the adversary is not allowed to forward queries from LF to F.

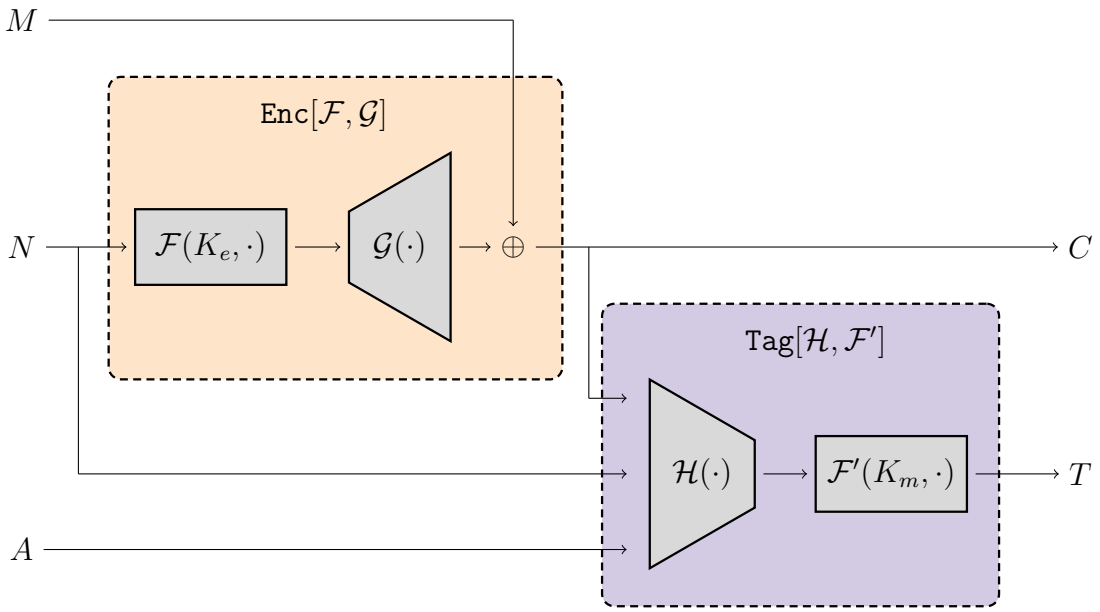

Figure 3.1: Graphical representation of the FGHF′ construction. It corresponds to the N2 composition of SE[$\mathcal{F}, \mathcal{G}$] and MAC[$\mathcal{H}, \mathcal{F}'$] which are in turn composed of a fixed-input-length function $\mathcal{F}$, a pseudorandom generator $\mathcal{G}$, a vector hash $\mathcal{H}$, and a fixed-input-length function $\mathcal{F}'$.

In fact, in the proof of Theorem 2.3.6 in [BMOS17], the SUF-CMLA adversary always forwards a particular query from its oracle LVfy to its oracle Vfy. This precludes reducing SUF-CMLA security to LPRF security due to our inability of simulating the verification oracles via the respective LPRF oracles. Note that Theorem 2.3.6 mandates SUF-CMLA security to be defined this way to hold, whereas lifting the restriction in the game LPRF would make it unsatisfiable. We overcome this problem by noting that, in the non-adaptive leakage setting, unpredictability suffices to achieve SUF-CMLA security, and at the same time we can allow the adversary to forward queries between its leakage and challenge oracles while maintaining satisfiability. This leads to a new security notion, called *leakage-resilient unpredictable function* (LUF), which we prove to be sufficient to yield SUF-CMLA security.

For both LUF and LPRF security we show that fixed-input-length functions are sufficient, as a collision-resistant vector hash function allows to turn fixed-input-length functions into variable-input-length functions while maintaining their LUF and LPRF security.

The advantage of the FGHF′ construction lies in its simplicity. It requires only two leakage-resilient functions which can be combined with off-the-shelf primitives for the pseudorandom generator $\mathcal{G}$ and hash function $\mathcal{H}$. Furthermore, for every encryption/decryption the leakage-resilient primitives are invoked only once.

Figure 3.2 illustrates the composition theorem for the FGHF′ construction that we develop in this section. It combines Theorem 2.3.6, developed by Barwell et al. [BMOS17], with the results we prove in the following sections.

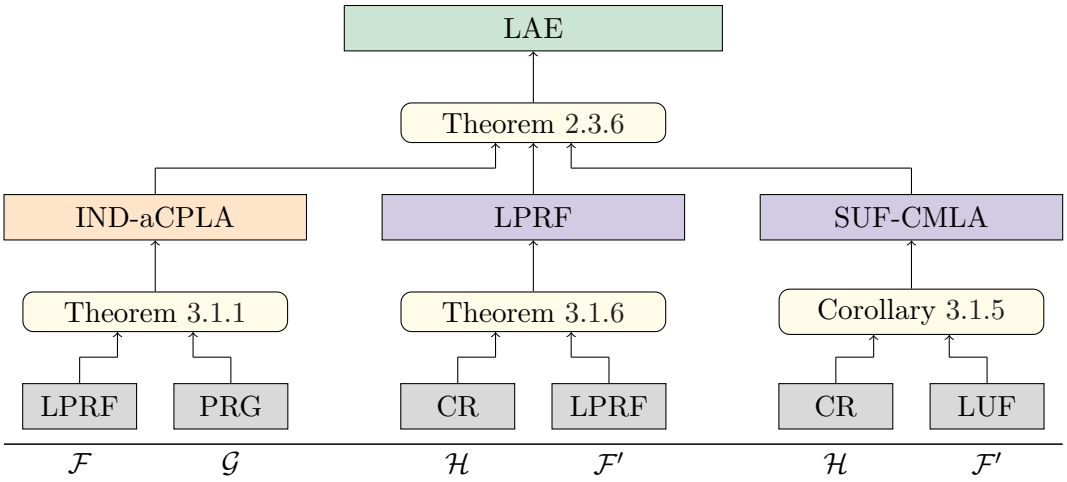

Figure 3.2: Visualisation of the FGHF′ composition theorem (cf. Theorem 3.1.7).

FGHF′-Enc$((K_e, K_m), N, A, M)$

// Compute ciphertext using SE$[\mathcal{F}, \mathcal{G}]$
$z \leftarrow \mathcal{F}(K_e, N)$
$C_e \leftarrow \mathcal{G}(z, \mu) \oplus M$
// Compute tag using MAC$[\mathcal{H}, \mathcal{F}']$
$H \leftarrow \mathcal{H}(N, A, C_e)$
$T \leftarrow \mathcal{F}'(K_m, H)$
return $C \leftarrow (C_e, T)$

FGHF′-Dec$((K_e, K_m), N, A, (C_e, T))$

$H \leftarrow \mathcal{H}(N, A, C_e)$
$T' \leftarrow \mathcal{F}'(K_m, H)$
if $T' \neq T$
  return $\perp$
$z \leftarrow \mathcal{F}(K_e, N)$
$M \leftarrow \mathcal{G}(z, |C_e|) \oplus C_e$
return M

Figure 3.3: Pseudocode of the FGHF′ construction with encryption and decryption algorithm FGHF′-Enc and FGHF′-Dec, respectively.

3.1.1 Leakage-Resilient Encryption $\text{SE}[\mathcal{F}, \mathcal{G}]$

The following theorem shows that the symmetric encryption scheme $\text{SE}[\mathcal{F}, \mathcal{G}]$ achieves IND-aCPLA security if the underlying components $\mathcal{F}$ and $\mathcal{G}$ achieve LPRF and PRG security, respectively. Note that security for this part even holds in the more general setting of adaptive leakage.

Theorem 3.1.1. *Let $\text{SE}[\mathcal{F}, \mathcal{G}]$ be the encryption scheme depicted in Figure 3.1, composed of the function family $\mathcal{F}\colon \mathcal{K} \times \{0,1\}^\nu \to \{0,1\}^n$ and the pseudorandom generator $\mathcal{G}\colon \{0,1\}^n \to \{0,1\}^*$ with respective associated leakage sets $\mathcal{L}_F$ and $\mathcal{L}_G$. Then for any* IND-aCPLA *adversary $\mathcal{A}_{\mathsf{se}}$ against $\text{SE}[\mathcal{F}, \mathcal{G}]$, making q queries to* Enc*, and associated leakage sets $\mathcal{L}_E = \mathcal{L}_D = \mathcal{L}_F \times \mathcal{L}_G$, there exists an* LPRF *adversary $\mathcal{A}_{\mathsf{lprf}}$ against $\mathcal{F}$ and a* PRG *adversary $\mathcal{A}_{\mathsf{prg}}$ against $\mathcal{G}$ such that*

$$\mathbf{Adv}^{\mathsf{INDaCPLA}}_{\text{SE}[\mathcal{F},\mathcal{G}]}(\mathcal{A}_{\mathsf{se}}, \mathcal{L}_E, \mathcal{L}_D) \leq \mathbf{Adv}^{\mathsf{LPRF}}_{\mathcal{F}}(\mathcal{A}_{\mathsf{lprf}}, \mathcal{L}_F) + q\,\mathbf{Adv}^{\mathsf{PRG}}_{\mathcal{G}}(\mathcal{A}_{\mathsf{prg}})\,.$$

Proof. The proof works through a sequence of games $\mathsf{G}_0, \mathsf{G}_1, \mathsf{G}_2, \mathsf{G}_3$ (cf. Figure 3.4). Game G_0 is the game INDaCPLA instantiated with $\text{SE}[\mathcal{F}, \mathcal{G}]$ and secret bit fixed to 1 and game G_3 is the same with secret bit fixed to 0. Using the equivalent notion of adversarial advantage, we obtain

$$\begin{aligned}
\mathbf{Adv}^{\mathsf{INDaCPLA}}_{\text{SE}[\mathcal{F},\mathcal{G}]}(\mathcal{A}_{\mathsf{se}}, \mathcal{L}_E) &= \left|2\,\Pr[\mathsf{INDaCPLA}^{\mathcal{A}_{\mathsf{se}}} \Rightarrow 1] - 1\right| \\
&= \left|\Pr[\mathcal{A}^{\mathsf{INDaCPLA}}_{\mathsf{se}} \Rightarrow 1 \mid b = 1] - \Pr[\mathcal{A}^{\mathsf{INDaCPLA}}_{\mathsf{se}} \Rightarrow 1 \mid b = 0]\right| \\
&= \left|\Pr[\mathcal{A}^{\mathsf{G}_0}_{\mathsf{se}} \Rightarrow 1] - \Pr[\mathcal{A}^{\mathsf{G}_3}_{\mathsf{se}} \Rightarrow 1]\right| \\
&\leq \sum_{i=1}^{3} \left|\Pr[\mathcal{A}^{\mathsf{G}_{i-1}}_{\mathsf{se}} \Rightarrow 1] - \Pr[\mathcal{A}^{\mathsf{G}_i}_{\mathsf{se}} \Rightarrow 1]\right|\,. \qquad (3.1)
\end{aligned}$$

We start with the first term for which we construct the following LPRF adversary $\mathcal{A}_{\mathsf{lprf}}$ against $\mathcal{F}$. For queries (N, M) to Enc by $\mathcal{A}_{\mathsf{se}}$, $\mathcal{A}_{\mathsf{lprf}}$ invokes its own challenge oracle on N to obtain z. Subsequently, it computes $Z \leftarrow \mathcal{G}(z, |M|)$ and sends $C \leftarrow Z \oplus M$ to $\mathcal{A}_{\mathsf{se}}$. Leakage queries $(N, M, (L_F, L_G))$ to LEnc are processed as follows. The tuple (z, Λ_F) is obtained from the oracle LF upon querying it on (N, L_F), while $C \leftarrow \mathcal{G}(z, |M|) \oplus M$ and $\Lambda_G \leftarrow L_G(z, M)$ are computed locally by $\mathcal{A}_{\mathsf{lprf}}$ before sending $(C, (\Lambda_F, \Lambda_G))$ to $\mathcal{A}_{\mathsf{se}}$. Leakage queries $(N, C, (L_F, L_G))$ to LDec are answered just as leakage queries to LEnc, except that L_G takes C, instead of M, as input and M is obtained by XORing C to the output of $\mathcal{G}$. When $\mathcal{A}$ outputs a bit b', $\mathcal{A}_{\mathsf{lprf}}$ forwards it as its own output.

Since the games G_0 and G_1 solely differ in generating the seed z for $\mathcal{G}$ during queries to Enc, $\mathcal{A}_{\mathsf{lprf}}$ perfectly simulates game G_0 and G_1 conditioned on the secret bit of the game LPRF being 1 and 0, respectively. We conclude with

$$\begin{aligned}
\left|\Pr[\mathcal{A}^{\mathsf{G}_0}_{\mathsf{se}} \Rightarrow 1] - \Pr[\mathcal{A}^{\mathsf{G}_1}_{\mathsf{se}} \Rightarrow 1]\right| &= \left|\Pr[\mathcal{A}^{\mathsf{LPRF}}_{\mathsf{lprf}} \Rightarrow 1 \mid b = 1] - \Pr[\mathcal{A}^{\mathsf{LPRF}}_{\mathsf{lprf}} \Rightarrow 1 \mid b = 0]\right| \\
&= \mathbf{Adv}^{\mathsf{LPRF}}_{\mathcal{F}}(\mathcal{A}_{\mathsf{lprf}}, \mathcal{L}_F)\,. \qquad (3.2)
\end{aligned}$$

Next, we bound the game hop between G_1 and G_2, for which we introduce a sequence of hybrid games $\mathsf{H}_0, \ldots, \mathsf{H}_q$, displayed in Figure 3.5. The games differ in how the value Z

is sampled. In H_i, for the first i queries, it is sampled ideally, i.e., a random bit string of length identical to the queried message. For the remaining $q - i$ queries, it is sampled real, i.e., the output of the PRG $\mathcal{G}$ on input a random seed and the length of the queried message. It follows that $\mathsf{H}_0 = \mathsf{G}_1$ and $\mathsf{H}_q = \mathsf{G}_2$, hence

$$\begin{aligned}\left|\Pr[\mathcal{A}_{\mathsf{se}}^{\mathsf{G}_1} \Rightarrow 1] - \Pr[\mathcal{A}_{\mathsf{se}}^{\mathsf{G}_2} \Rightarrow 1]\right| &= \left|\Pr[\mathcal{A}_{\mathsf{se}}^{\mathsf{H}_0} \Rightarrow 1] - \Pr[\mathcal{A}_{\mathsf{se}}^{\mathsf{H}_q} \Rightarrow 1]\right| \\ &\leq \sum_{i=1}^{q} \left|\Pr[\mathcal{A}_{\mathsf{se}}^{\mathsf{H}_{i-1}} \Rightarrow 1] - \Pr[\mathcal{A}_{\mathsf{se}}^{\mathsf{H}_i} \Rightarrow 1]\right| .\end{aligned}$$

We construct PRG adversaries $\mathcal{B}_1, \ldots, \mathcal{B}_q$ to bound the game hops between each pair of consecutive hybrid games. Adversary $\mathcal{B}_i$ proceeds as follows. It samples a key K for the function $\mathcal{F}$. This allows to perfectly simulate the leakage oracles LEnc and LDec for $\mathcal{A}_{\mathsf{se}}$. For leakage queries $(N, M, (L_F, L_G))$ to LEnc, it (locally) computes $z \leftarrow \mathcal{F}(K, N)$, $C \leftarrow \mathcal{G}(z, |M|) \oplus M$, $\Lambda_F \leftarrow L_F(K, N)$, and $\Lambda_G \leftarrow L_G(z, M)$, and sends $(C, (\Lambda_F, \Lambda_G))$ to $\mathcal{A}_{\mathsf{se}}$. Leakage queries $(N, C, (L_F, L_G))$ to LDec are processed similar, except that C is used instead of M for computing the leakage Λ_H. Let $(N_1, M_1), \ldots, (N_q, M_q)$ denote the challenge queries that $\mathcal{A}_{\mathsf{se}}$ makes to Enc. For queries (N_j, M_j) with $j < i$, $\mathcal{B}_i$ samples $Z \leftarrow\!\!\$\ \{0,1\}^{|M_j|}$ and sends $C \leftarrow Z \oplus M_j$ back to $\mathcal{A}_{\mathsf{se}}$. For queries (N_j, M_j) with $j > i$, $\mathcal{B}_i$ samples a seed z for $\mathcal{G}$, computes $Z \leftarrow \mathcal{G}(z, |M_j|)$ and sends $C \leftarrow Z \oplus M_j$ back to $\mathcal{A}_{\mathsf{se}}$. For the i^{th} query, (N_i, M_i), $\mathcal{B}_i$ invokes its own oracle G on $|M_i|$ to obtain Z, computes $C \leftarrow Z \oplus M_i$, and sends C back to $\mathcal{A}_{\mathsf{se}}$. It is easy to see that $\mathcal{B}_i$ simulates H_{i-1} and H_i if its challenge bit b from the game PRG is 1 and 0, respectively. Hence, we have

$$\begin{aligned}\left|\Pr[\mathcal{A}_{\mathsf{se}}^{\mathsf{H}_{i-1}} \Rightarrow 1] - \Pr[\mathcal{A}_{\mathsf{se}}^{\mathsf{H}_i} \Rightarrow 1]\right| &= \left|\Pr[\mathcal{B}_{\mathsf{i}}^{\mathsf{PRG}} \Rightarrow 1 \,|\, b = 1] - \Pr[\mathcal{B}_{\mathsf{i}}^{\mathsf{PRG}} \Rightarrow 1 \,|\, b = 1]\right| \\ &= \mathbf{Adv}_{\mathcal{G}}^{\mathsf{PRG}}(\mathcal{B}_{\mathsf{i}}) .\end{aligned}$$

We define $\mathcal{A}_{\mathsf{prg}}$ to be the adversary that picks $i \leftarrow\!\!\$\ [q]$ and then behaves as $\mathcal{B}_{\mathsf{i}}$. By a standard hybrid argument [FM21, MF21], we get

$$\begin{aligned}\left|\Pr[\mathcal{A}_{\mathsf{se}}^{\mathsf{G}_1} \Rightarrow 1] - \Pr[\mathcal{A}_{\mathsf{se}}^{\mathsf{G}_2} \Rightarrow 1]\right| &\leq \sum_{i=1}^{q} \left|\Pr[\mathcal{A}_{\mathsf{se}}^{\mathsf{H}_{i-1}} \Rightarrow 1] - \Pr[\mathcal{A}_{\mathsf{se}}^{\mathsf{H}_i} \Rightarrow 1]\right| \\ &\leq q\, \mathbf{Adv}_{\mathcal{G}}^{\mathsf{PRG}}(\mathcal{A}_{\mathsf{prg}}) . \qquad (3.3)\end{aligned}$$

Since the challenge oracle Enc outputs identically distributed ciphertexts in G_2 and G_3 the adversary can not distinguish these games which yields

$$\left|\Pr[\mathcal{A}_{\mathsf{se}}^{\mathsf{G}_2} \Rightarrow 1] - \Pr[\mathcal{A}_{\mathsf{se}}^{\mathsf{G}_3} \Rightarrow 1]\right| = 0 . \qquad (3.4)$$

Inserting (3.2), (3.3), and (3.4) into (3.1) yields

$$\mathbf{Adv}_{\mathrm{SE}[\mathcal{F},\mathcal{G}]}^{\mathsf{INDaCPLA}}(\mathcal{A}_{\mathsf{se}}, \mathcal{L}_E, \mathcal{L}_D) \leq \mathbf{Adv}_{\mathcal{F}}^{\mathsf{LPRF}}(\mathcal{A}_{\mathsf{lprf}}, \mathcal{L}_F) + q\, \mathbf{Adv}_{\mathcal{G}}^{\mathsf{PRG}}(\mathcal{A}_{\mathsf{prg}}) .$$

This proves the claim. □

Games $\mathsf{G}_0, \mathsf{G}_1, \mathsf{G}_2, \mathsf{G}_3$

$K \leftarrow_{\$} \mathcal{K}$
$b' \leftarrow \mathcal{A}^{\mathsf{Enc},\mathsf{LEnc},\mathsf{LDec}}()$

oracle $\mathsf{Enc}(N, M)$ in G_0

$z \leftarrow \mathcal{F}(K, N)$
$Z \leftarrow \mathcal{G}(z, \mu)$
return $C \leftarrow Z \oplus M$

oracle $\mathsf{Enc}(N, M)$ in G_1

$z \leftarrow_{\$} \{0,1\}^n$
$Z \leftarrow \mathcal{G}(z, \mu)$
return $C \leftarrow Z \oplus M$

oracle $\mathsf{Enc}(N, M)$ in G_2

$Z \leftarrow_{\$} \{0,1\}^\mu$
return $C \leftarrow Z \oplus M$

oracle $\mathsf{Enc}(N, M)$ in G_3

return $C \leftarrow_{\$} \{0,1\}^\mu$

oracle $\mathsf{LEnc}(N, M, (L_F, L_G))$

$z \leftarrow \mathcal{F}(K, N)$
$Z \leftarrow \mathcal{G}(z, \mu)$
$C \leftarrow Z \oplus M$
$\Lambda_F \leftarrow L_F(K, N)$
$\Lambda_G \leftarrow L_G(z, M)$
return $(C, (\Lambda_F, \Lambda_G))$

oracle $\mathsf{LDec}(N, C, (L_F, L_G))$

$z \leftarrow \mathcal{F}(K, N)$
$Z \leftarrow \mathcal{G}(z, \gamma)$
$M \leftarrow Z \oplus C$
$\Lambda_F \leftarrow L_F(K, N)$
$\Lambda_G \leftarrow L_G(z, C)$
return $(M, (\Lambda_F, \Lambda_G))$

Figure 3.4: Games G_0, G_1, G_2, and G_3 used in the proof of Theorem 3.1.1. The oracles LEnc and LDec are shared across all games. Each game uses the oracle Enc as specified.

Game H_i

$K \leftarrow_{\$} \mathcal{K}$
$c \leftarrow 0$
$b' \leftarrow \mathcal{A}^{\mathsf{Enc},\mathsf{LEnc},\mathsf{LDec}}()$

oracle $\mathsf{Enc}(N, M)$ in H_i

$c \leftarrow c + 1$
if $c \leq i$
  $Z \leftarrow_{\$} \{0,1\}^\mu$
else
  $z \leftarrow_{\$} \{0,1\}^n$
  $Z \leftarrow \mathcal{G}(z, \mu)$
return $C \leftarrow Z \oplus M$

oracle $\mathsf{LEnc}(N, M, (L_F, L_G))$

$z \leftarrow \mathcal{F}(K, N)$
$Z \leftarrow \mathcal{G}(z, \mu)$
$C \leftarrow Z \oplus M$
$\Lambda_F \leftarrow L_F(K, N)$
$\Lambda_G \leftarrow L_G(z, M)$
return $(C, (\Lambda_F, \Lambda_G))$

oracle $\mathsf{LDec}(N, C, (L_F, L_G))$

$z \leftarrow \mathcal{F}(K, N)$
$Z \leftarrow \mathcal{G}(z, \gamma)$
$M \leftarrow Z \oplus C$
$\Lambda_F \leftarrow L_F(K, N)$
$\Lambda_G \leftarrow L_G(z, C)$
return $(M, (\Lambda_F, \Lambda_G))$

Figure 3.5: Hybrid games H_i used in the proof of Theorem 3.1.1. The games share the leakage oracles LEnc and LDec and differ only in the oracle Enc.

3.1.2 Leakage-Resilient Message Authentication $\textsc{Mac}[\mathcal{H}, \mathcal{F}']$

As explained above, LPRF security is insufficient to achieve SUF-CMLA security. To circumvent this issue, we introduce the notion of a leakage-resilient unpredictable function (LUF). We then show that SUF-CMLA security follows from LUF security for the canonical MAC construction. To deal with messages of arbitrary length, we show further that the domain of a fixed-input-length LUF can be extended via a collision-resistant hash function. These two results then yield the SUF-CMLA security of $\textsc{Mac}[\mathcal{H}, \mathcal{F}']$. The final piece is to prove that the domain extension using a collision-resistant hash function also works for a fixed-input-length LPRF, in order to show the tagging algorithm of $\textsc{Mac}[\mathcal{H}, \mathcal{F}']$ to be an LPRF.

Leakage-Resilient Unpredictable Functions

The definition below defines leakage-resilient unpredictable functions. Note that, unlike for LPRF security where the adversary must not forward queries from its leakage oracle to its challenge oracle, LUF security allows the adversary to obtain leakage for the input for which it tries to predict the output. However, note further that the leakage oracle only outputs the leakage but not the output of the function.

Definition 3.1.2 (LUF Security)**.** *Let $\mathcal{F}\colon \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ be a function family over the domain $\mathcal{X}$ and indexed by $\mathcal{K}$, and the game* LUF *be as defined in Figure 3.6. Then, for any adversary $\mathcal{A}$ its corresponding* LUF *advantage is given by*

$$\mathbf{Adv}^{\mathsf{LUF}}_{\mathcal{F}}(\mathcal{A}, \mathcal{L}_F) \coloneqq \Pr[\mathsf{LUF}^{\mathcal{A}} \Rightarrow 1]\,.$$

Game LUF	**oracle** $\mathsf{Guess}(X, Y')$	**oracle** $\mathsf{F}(X)$	**oracle** $\mathsf{Lkg}(X, L)$
$\mathsf{win} \leftarrow 0$	$Y \leftarrow \mathcal{F}(K, X)$	$\mathcal{S} \leftarrow_{\cup} X$	$\Lambda \leftarrow L(K, X)$
$\mathcal{S} \leftarrow \emptyset$	**if** $X \notin \mathcal{S} \wedge Y = Y'$	$Y \leftarrow \mathcal{F}(K, X)$	**return** Λ
$K \leftarrow_{\$} \mathcal{K}$	$\quad \mathsf{win} \leftarrow 1$	**return** Y	
$\mathcal{A}^{\mathsf{Guess},\mathsf{F},\mathsf{Lkg}}()$	**return** $(Y = Y')$		
return win			

Figure 3.6: Security game LUF.

Unpredictability of $\textsc{Mac}[\mathcal{H}, \mathcal{F}']$

The following theorem shows that LUF security implies SUF-CMLA security via the canonical MAC construction.

Theorem 3.1.3. *Let $\hat{\mathcal{F}}\colon \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ be a function family with associated leakage set $\mathcal{L}_{\hat{\mathcal{F}}}$, and let $\textsc{Mac}[\hat{\mathcal{F}}]$ be the corresponding canonical MAC with associated leakage sets $\mathcal{L}_T$, $\mathcal{L}_V$ where $\mathcal{L}_T = \mathcal{L}_V = \mathcal{L}_{\hat{\mathcal{F}}}$. Then for any* SUF-CMLA *adversary $\mathcal{A}_{\mathsf{mac}}$ against $\textsc{Mac}[\hat{\mathcal{F}}]$, there exists an adversary $\mathcal{A}_{\mathsf{luf}}$ against $\hat{\mathcal{F}}$ such that*

$$\mathbf{Adv}^{\mathsf{SUFCMLA}}_{\textsc{Mac}[\hat{\mathcal{F}}]}(\mathcal{A}_{\mathsf{mac}}, \mathcal{L}_T, \mathcal{L}_V) \leq \mathbf{Adv}^{\mathsf{LUF}}_{\hat{\mathcal{F}}}(\mathcal{A}_{\mathsf{luf}}, \mathcal{L}_{\hat{F}})\,.$$

Proof. For any SUF-CMLA adversary $\mathcal{A}_{\mathsf{mac}}$ against MAC we construct a LUF adversary $\mathcal{A}_{\mathsf{luf}}$ against $\hat{\mathcal{F}}$. We proceed in two steps, we first bound the SUF-CMLA advantage of $\mathcal{A}_{\mathsf{mac}}$ by the probability of a particular event occurring and then we bound this probability by the LUF advantage of $\mathcal{A}_{\mathsf{luf}}$. Let E be the event that $\mathcal{A}_{\mathsf{mac}}$ makes a query to its oracle Vfy which returns $\top$. Then

$$\begin{aligned}\mathbf{Adv}^{\mathsf{SUFCMLA}}_{\mathrm{MAC}[\hat{\mathcal{F}}]}(\mathcal{A}_{\mathsf{mac}}, \mathcal{L}_T, \mathcal{L}_V) &= \left|2\ \Pr[\mathsf{SUFCMLA}^{\mathcal{A}_{\mathsf{mac}}} \Rightarrow 1] - 1\right| \\ &= \left|2\ \Pr[\mathsf{SUFCMLA}^{\mathcal{A}_{\mathsf{mac}}} \Rightarrow 1 \cap E]\right. \\ &\quad \left. + 2\ \Pr[\mathsf{SUFCMLA}^{\mathcal{A}_{\mathsf{mac}}} \Rightarrow 1 \cap \overline{E}] - 1\right| \\ &\leq \left|2\ \Pr[E] + 2\ \Pr[\mathsf{SUFCMLA}^{\mathcal{A}_{\mathsf{mac}}} \Rightarrow 1 \,|\, \overline{E}] - 1\right| . \qquad (3.5)\end{aligned}$$

Now, conditioned on $\overline{E}$ the oracle Vfy will always return $\perp$ irrespective of the value of b and hence the probability of $\mathcal{A}_{\mathsf{mac}}$ winning is exactly one half. Thus the above can be re-written as

$$\begin{aligned}\left|2\ \Pr[E] + 2\Pr[\mathsf{SUFCMLA}^{\mathcal{A}_{\mathsf{mac}}} \Rightarrow 1 \,|\, \overline{E}] - 1\right| &= 2\ \Pr[E] \\ &= 2\ \Pr[E \cap b = 0] + 2\Pr[E \cap b = 1] . \qquad (3.6)\end{aligned}$$

When $b = 0$ the oracle Vfy will always return $\perp$ and thus E simply cannot occur, i.e., the first term in the above expression is zero. Thus

$$\begin{aligned}2\ \Pr[E \cap b = 0] + 2\Pr[E \cap b = 1] &= 2\ \Pr[E \,|\, b = 1] \Pr[b = 1] \\ &= \Pr[E \,|\, b = 1] . \qquad (3.7)\end{aligned}$$

We now bound this last probability by the LUF advantage of $\mathcal{A}_{\mathsf{luf}}$. By construction we have that $\mathcal{L}_T = \mathcal{L}_V = \mathcal{L}_{\hat{F}}$. Then $\mathcal{A}_{\mathsf{luf}}$ runs $\mathcal{A}_{\mathsf{mac}}$ and provides it with a simulation of the game $\mathsf{SUFCMLA}$ with the bit b fixed to 1. Whenever $\mathcal{A}_{\mathsf{mac}}$ makes a query (X, L) to LTag, $\mathcal{A}_{\mathsf{luf}}$ queries X to F and (X, L) to Lkg to obtain $Y = \hat{\mathcal{F}}(K, X)$ and $\Lambda = L(K, X)$, respectively, and returns (Y, Λ) back to $\mathcal{A}_{\mathsf{mac}}$. Similarly if $\mathcal{A}_{\mathsf{mac}}$ queries (X, T, L) to its LVfy oracle, $\mathcal{A}_{\mathsf{luf}}$ queries (X, L) to Lkg and (X, T) to Guess to obtain Λ and V, respectively, and returns (V, Λ) to $\mathcal{A}_{\mathsf{mac}}$. Every query (X, T) that $\mathcal{A}_{\mathsf{mac}}$ makes to its oracle Vfy, is forwarded by $\mathcal{A}_{\mathsf{luf}}$ to its own oracle Guess and returns $\top$ to $\mathcal{A}_{\mathsf{mac}}$ if Guess returns 1 and $\perp$ otherwise.

Recall that $\mathcal{A}_{\mathsf{luf}}$'s queries to F are recorded in the set $\mathcal{S}$, and it only wins the game LUF if it makes a query to Guess which returns 1 where the corresponding X value is not contained in $\mathcal{S}$. However since $\mathcal{A}_{\mathsf{mac}}$ does not forward queries from LTag to Vfy, it is guaranteed that $\mathcal{A}_{\mathsf{luf}}$'s queries to Guess are not contained in $\mathcal{S}$. It then follows that $\mathcal{A}_{\mathsf{luf}}$ wins the game LUF whenever E occurs, and hence

$$\Pr[E \,|\, b = 1] \leq \mathbf{Adv}^{\mathsf{LUF}}_{\hat{\mathcal{F}}}(\mathcal{A}_{\mathsf{luf}}, \mathcal{L}_{\hat{F}}) . \qquad (3.8)$$

By combining (3.5), (3.6), (3.7), and (3.8), we obtain

$$\mathbf{Adv}^{\mathsf{SUFCMLA}}_{\mathrm{MAC}[\hat{\mathcal{F}}]}(\mathcal{A}_{\mathsf{mac}}, \mathcal{L}_T, \mathcal{L}_V) \leq \mathbf{Adv}^{\mathsf{LUF}}_{\hat{\mathcal{F}}}(\mathcal{A}_{\mathsf{luf}}, \mathcal{L}_{\hat{F}}) ,$$

which proves the theorem. □

The theorem above only yields SUF-CMLA secure MACs with the same domain as the underlying function. The following theorem shows that this is not a hindrance as the domain of the underlying function can be extended by a collision-resistant hash function while maintaining its LUF security.

Theorem 3.1.4. *Let $\mathcal{F}'\colon \mathcal{K} \times \{0,1\}^w \to \mathcal{Y}$ be a function family with associated leakage set $\mathcal{L}_{F'}$, and let $\mathcal{H}\colon \mathcal{X} \to \{0,1\}^w$ be a hash function over any domain $\mathcal{X}$. Further, let their composition $\hat{\mathcal{F}}$ be defined as*

$$\hat{\mathcal{F}}(K, X) = \mathcal{F}'(K, \mathcal{H}(X))\,,$$

where $X \in \mathcal{X}$, $K \in \mathcal{K}$, and $\mathcal{L}_{\hat{F}} = \mathcal{L}_{F'} \times \mathcal{L}_H$ for any set of efficiently computable functions $\mathcal{L}_H$. Then for any LUF *adversary $\mathcal{A}_{\mathsf{luf}}$ against $\hat{\mathcal{F}}$, there exists a corresponding* LUF *adversary $\overline{\mathcal{A}}_{\mathsf{luf}}$ against $\mathcal{F}'$ and an adversary $\mathcal{A}_{\mathsf{hash}}$ against $\mathcal{H}$ such that*

$$\mathbf{Adv}^{\mathsf{LUF}}_{\hat{\mathcal{F}}}(\mathcal{A}_{\mathsf{luf}}, \mathcal{L}_{\hat{F}}) \leq \mathbf{Adv}^{\mathsf{CR}}_{\mathcal{H}}(\mathcal{A}_{\mathsf{hash}}) + \mathbf{Adv}^{\mathsf{LUF}}_{\mathcal{F}'}(\overline{\mathcal{A}}_{\mathsf{luf}}, \mathcal{L}_{F'})\,.$$

Proof. We prove the theorem using game G displayed in Figure 3.7. Except for some additional bookkeeping, game G equals LUF and $\boxed{\mathsf{G}}$ is a variant where queries which form a collision in the hash function with a previously made query are answered with $\perp$. It holds that

$$\begin{aligned}\mathbf{Adv}^{\mathsf{LUF}}_{\hat{\mathcal{F}}}(\mathcal{A}_{\mathsf{luf}}, \mathcal{L}_{\hat{F}}) &= \Pr[\mathsf{G}^{\mathcal{A}_{\mathsf{luf}}} \Rightarrow 1] \\ &= \Pr[\mathsf{G}^{\mathcal{A}_{\mathsf{luf}}} \Rightarrow 1] - \Pr[\boxed{\mathsf{G}}^{\mathcal{A}_{\mathsf{luf}}} \Rightarrow 1] + \Pr[\boxed{\mathsf{G}}^{\mathcal{A}_{\mathsf{luf}}} \Rightarrow 1]\,. \qquad (3.9)\end{aligned}$$

From the games it is easy to see that G and $\boxed{\mathsf{G}}$ are identical until the flag Bad is set which leads to

$$\Pr[\mathsf{G}^{\mathcal{A}_{\mathsf{luf}}} \Rightarrow 1] - \Pr[\boxed{\mathsf{G}}^{\mathcal{A}_{\mathsf{luf}}} \Rightarrow 1] \leq \Pr[\mathcal{A}^{\mathsf{G}}_{\mathsf{luf}} \text{ sets } \mathsf{Bad}]\,. \qquad (3.10)$$

We construct the following adversary $\mathcal{A}_{\mathsf{hash}}$. It chooses a key K for $\mathcal{F}'$ at random. For every query (X, Y') to Guess, $\mathcal{A}_{\mathsf{hash}}$ computes $H \leftarrow \mathcal{H}(X)$ and $Y \leftarrow \mathcal{F}'(K, H)$. It returns 1, if $Y = Y'$, otherwise, it returns 0. For queries X to F, $\mathcal{A}_{\mathsf{hash}}$ computes $H \leftarrow \mathcal{H}(X)$ as well as $Y \leftarrow \mathcal{F}'(K, H)$ and sends the latter back to $\mathcal{A}_{\mathsf{luf}}$.[16] When $\mathcal{A}_{\mathsf{luf}}$ makes a query $(X, (L_H, L_{F'}))$ to Lkg, $\mathcal{A}_{\mathsf{hash}}$ computes $H \leftarrow \mathcal{H}(X)$, $\Lambda_H \leftarrow L_H(X)$, and $\Lambda_{F'} \leftarrow L_{F'}(K, H)$. Then it sends $(\Lambda_H, \Lambda_{F'})$ back to $\mathcal{A}_{\mathsf{luf}}$. Throughout, $\mathcal{A}_{\mathsf{hash}}$ monitors the hash values that stem from $\mathcal{A}_{\mathsf{luf}}$'s queries. If a query X, either to Guess or F, triggers Bad, $\mathcal{A}_{\mathsf{hash}}$ looks up the colliding input X' and outputs the collision (X, X'). We conclude with

$$\Pr[\mathcal{A}^{\mathsf{G}}_{\mathsf{luf}} \text{ sets } \mathsf{Bad}] \leq \mathbf{Adv}^{\mathsf{CR}}_{\mathcal{H}}(\mathcal{A}_{\mathsf{hash}})\,. \qquad (3.11)$$

To bound the probability that $\mathcal{A}_{\mathsf{luf}}$ wins game $\boxed{\mathsf{G}}$, we construct another adversary $\overline{\mathcal{A}}_{\mathsf{luf}}$ against $\mathcal{F}'$. Specifically, $\overline{\mathcal{A}}_{\mathsf{luf}}$ runs $\mathcal{A}_{\mathsf{luf}}$ and answers queries as follows. Leakage queries $(X, (L_H, L_{F'}))$ are answered by locally computing $H \leftarrow \mathcal{H}(X)$ and $\Lambda_H \leftarrow L_H(X)$, querying $(H, L_{F'})$ to its own leakage oracle to obtain $\Lambda_{F'}$, and sending $(\Lambda_H, \Lambda_{F'})$ back to $\mathcal{A}_{\mathsf{luf}}$. For

[16] For simplicity we ignore the winning condition as $\mathcal{A}_{\mathsf{hash}}$ solely aims to find a collision for the hash function.

queries X and (X, Y') to F and Guess, respectively, $\overline{\mathcal{A}}_{\mathsf{luf}}$ forwards the query, replacing X with the corresponding hash value $H \leftarrow \mathcal{H}(X)$, to its own oracle and passes the response back to $\mathcal{A}_{\mathsf{luf}}$. However, if the hash value H has already occurred during a prior query by $\mathcal{A}_{\mathsf{luf}}$, $\overline{\mathcal{A}}_{\mathsf{luf}}$ simply responds with $\perp$ without making a query itself. It remains to argue that $\overline{\mathcal{A}}_{\mathsf{luf}}$ wins whenever $\mathcal{A}_{\mathsf{luf}}$ wins. This follows from the simple observation that the winning query by $\mathcal{A}_{\mathsf{luf}}$, i.e., the one that set win to true, must be on an input resulting in a fresh hash value. Since $\overline{\mathcal{A}}_{\mathsf{luf}}$ forwards the hash value to its own oracle, this hash value has not been queried to F, hence, it will also set win to true. We conclude with

$$\Pr[\boxed{\mathsf{G}}^{\mathcal{A}_{\mathsf{luf}}} \Rightarrow 1] \leq \mathbf{Adv}^{\mathsf{LUF}}_{\mathcal{F}'}(\overline{\mathcal{A}}_{\mathsf{luf}}, \mathcal{L}_{F'}) \,. \tag{3.12}$$

Collecting (3.10), (3.11), and (3.12), and inserting into (3.9) yields

$$\begin{aligned}\mathbf{Adv}^{\mathsf{LUF}}_{\hat{\mathcal{F}}}(\mathcal{A}_{\mathsf{luf}}, \mathcal{L}_{\hat{F}}) &= \Pr[\mathsf{G}^{\mathcal{A}_{\mathsf{luf}}} \Rightarrow 1] - \Pr[\boxed{\mathsf{G}}^{\mathcal{A}_{\mathsf{luf}}} \Rightarrow 1] + \Pr[\boxed{\mathsf{G}}^{\mathcal{A}_{\mathsf{luf}}} \Rightarrow 1] \\ &\leq \mathbf{Adv}^{\mathsf{CR}}_{\mathcal{H}}(\mathcal{A}_{\mathsf{hash}}) + \mathbf{Adv}^{\mathsf{LUF}}_{\mathcal{F}'}(\overline{\mathcal{A}}_{\mathsf{luf}}, \mathcal{L}_{F'}) \,,\end{aligned}$$

which concludes the proof. □

Games $\mathsf{G}, \boxed{\mathsf{G}}$	**oracle** $\mathsf{Guess}(X, Y')$	**oracle** $\mathsf{F}(X)$	**oracle** $\mathsf{Lkg}(X, (L_H, L_{F'}))$
$\mathsf{win} \leftarrow 0$	$H \leftarrow \mathcal{H}(X)$	$H \leftarrow \mathcal{H}(X)$	$H \leftarrow \mathcal{H}(X)$
$\mathcal{S} \leftarrow \emptyset$	**if** $H \in \mathcal{T}$	**if** $H \in \mathcal{T}$	$\Lambda_H \leftarrow L_H(X)$
$\mathcal{T} \leftarrow \emptyset$	$\quad \mathsf{Bad} \leftarrow \mathrm{true}$	$\quad \mathsf{Bad} \leftarrow \mathrm{true}$	$\Lambda_{F'} \leftarrow L_{F'}(K, H)$
$\mathsf{Bad} \leftarrow \mathrm{false}$	$\quad \boxed{\textbf{return } \perp}$	$\quad \boxed{\textbf{return } \perp}$	$\Lambda \leftarrow (\Lambda_H, \Lambda_{F'})$
$K \leftarrow\!\!\$\, \mathcal{K}$	$\mathcal{T} \leftarrow\!\!\cup\, H$	$\mathcal{T} \leftarrow\!\!\cup\, H$	**return** Λ
$\mathcal{A}^{\mathsf{Guess},\mathsf{F},\mathsf{Lkg}}()$	$Y \leftarrow \mathcal{F}'(K, H)$	$\mathcal{S} \leftarrow\!\!\cup\, X$	
return win	**if** $X \notin \mathcal{S} \wedge Y = Y'$	$Y \leftarrow \mathcal{F}'(K, H)$	
	$\quad \mathsf{win} \leftarrow 1$	**return** Y	
	return $(Y = Y')$		

Figure 3.7: Games used in the proof of Theorem 3.1.4.

Combining Theorem 3.1.3 and Theorem 3.1.4, we obtain the following corollary, which reduces the SUF-CMLA security of $\textsc{Mac}[\mathcal{H}, \mathcal{F}']$ to that of its building blocks $\mathcal{H}$ and $\mathcal{F}'$.

Corollary 3.1.5. *Let* $\textsc{Mac}[\mathcal{H}, \mathcal{F}']$ *be the MAC depicted in Figure 3.1, composed of the hash function* $\mathcal{H}$ *and the function family* $\mathcal{F}'$ *with respective leakage sets* $\mathcal{L}_H$ *and* $\mathcal{L}_{F'}$*. Then, for any* SUF-CMLA *adversary* $\mathcal{A}_{\mathsf{mac}}$ *against* $\textsc{Mac}[\mathcal{H}, \mathcal{F}']$ *with associated leakage sets* $\mathcal{L}_T = \mathcal{L}_V = \mathcal{L}_H \times \mathcal{L}_{F'}$*, there exists a* LUF *adversary* $\mathcal{A}_{\mathsf{luf}}$ *against* $\mathcal{F}'$ *and an adversary* $\mathcal{A}_{\mathsf{hash}}$ *against* $\mathcal{H}$ *such that*

$$\mathbf{Adv}^{\mathsf{SUFCMLA}}_{\textsc{Mac}[\mathcal{H},\mathcal{F}']}(\mathcal{A}_{\mathsf{mac}}, \mathcal{L}_T, \mathcal{L}_V) \leq \mathbf{Adv}^{\mathsf{CR}}_{\mathcal{H}}(\mathcal{A}_{\mathsf{hash}}) + \mathbf{Adv}^{\mathsf{LUF}}_{\mathcal{F}'}(\mathcal{A}_{\mathsf{luf}}, \mathcal{L}_{F'}) \,.$$

Proof. It holds that

$$\begin{aligned}\mathbf{Adv}^{\mathsf{SUFCMLA}}_{\textsc{Mac}[\mathcal{H},\mathcal{F}']}(\mathcal{A}_{\mathsf{mac}}, \mathcal{L}_T, \mathcal{L}_V) &\overset{(\textit{Theorem } 3.1.3)}{\leq} \mathbf{Adv}^{\mathsf{LUF}}_{\hat{\mathcal{F}}}(\overline{\mathcal{A}}_{\mathsf{luf}}, \mathcal{L}_{\hat{F}}) \\ &\overset{(\textit{Theorem } 3.1.4)}{\leq} \mathbf{Adv}^{\mathsf{CR}}_{\mathcal{H}}(\mathcal{A}_{\mathsf{hash}}) + \mathbf{Adv}^{\mathsf{LUF}}_{\mathcal{F}'}(\mathcal{A}_{\mathsf{luf}}, \mathcal{L}_{F'}) \,,\end{aligned}$$

where $\hat{\mathcal{F}}$ is the composition of $\mathcal{H}$ and $\mathcal{F}'$ with associated leakage set $\mathcal{L}_{\hat{F}} = \mathcal{L}_H \times \mathcal{L}_{F'}$. This proves the claim. □

Pseudorandomness of $\textsc{Mac}[\mathcal{H}, \mathcal{F}']$

The final piece needed to apply Theorem 2.3.6 is to show that $\textsc{Mac}[\mathcal{H}, \mathcal{F}']$, or rather its tagging algorithm $\mathtt{Tag}[\mathcal{H}, \mathcal{F}']$, is a leakage-resilient pseudorandom function. Since by assumption $\mathcal{F}'$ is already an LPRF, this result is analogous to Theorem 3.1.4 in that it provides us a simple technique for extending the domain of an LPRF with a collision-resistant hash function.

Theorem 3.1.6. *Let $\mathcal{F}' \colon \mathcal{K} \times \{0,1\}^w \to \mathcal{Y}$ be a function family with associated leakage set $\mathcal{L}_{F'}$, and let $\mathcal{H} \colon \mathcal{X} \to \{0,1\}^w$ be a hash function over the domain $\mathcal{X}$. Further, let their composition $\hat{\mathcal{F}}$ be defined by*

$$\hat{\mathcal{F}}(K, X) = \mathcal{F}'(K, \mathcal{H}(X)) \,,$$

where $X \in \mathcal{X}$, $K \in \mathcal{K}$, and $\mathcal{L}_{\hat{F}} = \mathcal{L}_H \times \mathcal{L}_{F'}$ for any set of efficiently computable functions $\mathcal{L}_H$. Then, for any LPRF *adversary $\mathcal{A}_{\mathsf{lprf}}$ against $\hat{\mathcal{F}}$, there exists a corresponding* LPRF *adversary $\overline{\mathcal{A}}_{\mathsf{lprf}}$ against $\mathcal{F}'$ and an adversary $\mathcal{A}_{\mathsf{hash}}$ against $\mathcal{H}$ such that*

$$\mathbf{Adv}^{\mathsf{LPRF}}_{\hat{\mathcal{F}}}(\mathcal{A}_{\mathsf{lprf}}, \mathcal{L}_{\hat{F}}) \leq 2\,\mathbf{Adv}^{\mathsf{CR}}_{\mathcal{H}}(\mathcal{A}_{\mathsf{hash}}) + \mathbf{Adv}^{\mathsf{LPRF}}_{\mathcal{F}'}(\overline{\mathcal{A}}_{\mathsf{lprf}}, \mathcal{L}_{F'}) \,.$$

Proof. We prove the theorem using game G displayed in Figure 3.8. It holds that game G is the game LPRF, except for some additional bookkeeping. Game $\boxed{\mathsf{G}}$ is similar, with the exception that only queries with a fresh hash value are answered.

$$\begin{aligned}
\mathbf{Adv}^{\mathsf{LPRF}}_{\hat{\mathcal{F}}}(\mathcal{A}_{\mathsf{lprf}}, \mathcal{L}_{\hat{F}}) &= \mathbf{Adv}^{\mathsf{G}}(\mathcal{A}_{\mathsf{lprf}}, \mathcal{L}_{\hat{F}}) \\
&= \Big|\Pr[\mathcal{A}^{\mathsf{G}}_{\mathsf{lprf}} \Rightarrow 1 \,|\, b = 1] - \Pr[\mathcal{A}^{\mathsf{G}}_{\mathsf{lprf}} \Rightarrow 1 \,|\, b = 0]\Big| \\
&\leq \Big|\Pr[\mathcal{A}^{\mathsf{G}}_{\mathsf{lprf}} \Rightarrow 1 \,|\, b = 1] - \Pr[\mathcal{A}^{\boxed{\mathsf{G}}}_{\mathsf{lprf}} \Rightarrow 1 \,|\, b = 1]\Big| \\
&\quad + \Big|\Pr[\mathcal{A}^{\boxed{\mathsf{G}}}_{\mathsf{lprf}} \Rightarrow 1 \,|\, b = 1] - \Pr[\mathcal{A}^{\boxed{\mathsf{G}}}_{\mathsf{lprf}} \Rightarrow 1 \,|\, b = 0]\Big| \\
&\quad + \Big|\Pr[\mathcal{A}^{\boxed{\mathsf{G}}}_{\mathsf{lprf}} \Rightarrow 1 \,|\, b = 0] - \Pr[\mathcal{A}^{\mathsf{G}}_{\mathsf{lprf}} \Rightarrow 1 \,|\, b = 0]\Big| \,. \qquad (3.13)
\end{aligned}$$

We start with the first difference, that is, between game G and game $\boxed{\mathsf{G}}$ with secret bit fixed to 1. It is easy to see that

$$\Big|\Pr[\mathcal{A}^{\mathsf{G}}_{\mathsf{lprf}} \Rightarrow 1 \,|\, b = 1] - \Pr[\mathcal{A}^{\boxed{\mathsf{G}}}_{\mathsf{lprf}} \Rightarrow 1 \,|\, b = 1]\Big| \leq \Pr[\mathcal{A}^{\mathsf{G}}_{\mathsf{lprf}} \text{ sets } \mathsf{Bad} \,|\, b = 1] \,. \qquad (3.14)$$

We construct the following adversary $\mathcal{A}_1$ against $\mathcal{H}$. It chooses a key K for $\mathcal{F}'$ at random. For every query X to F by $\mathcal{A}_{\mathsf{lprf}}$, $\mathcal{A}_1$ computes $Y \leftarrow \mathcal{F}'(K, \mathcal{H}(X))$ and sends it back to $\mathcal{A}_{\mathsf{lprf}}$. For leakage queries $(X, (L_H, L_{F'}))$ to LF, $\mathcal{A}_1$ computes $Y \leftarrow \mathcal{F}'(K, \mathcal{H}(X))$, $\Lambda_H \leftarrow L_H(X)$, and $\Lambda_{F'} \leftarrow L_{F'}(K, H)$. Then it sends $(Y, (\Lambda_H, \Lambda_{F'}))$ back to $\mathcal{A}_{\mathsf{lprf}}$. For all these queries, $\mathcal{A}_1$ monitors the hash values H that occur while evaluating $\mathcal{A}_{\mathsf{lprf}}$'s queries. Upon observing a collision, i.e., $\mathcal{A}_{\mathsf{lprf}}$ makes a query X that triggers Bad, $\mathcal{A}_1$ looks up the prior query X'

that resulted in the same hash value and outputs (X, X') as a collision for $\mathcal{H}$. It follows that

$$\Pr[\mathcal{A}_{\mathsf{lprf}}^{\mathsf{G}} \text{ sets } \mathsf{Bad} \mid b = 1] = \mathbf{Adv}_{\mathcal{H}}^{\mathsf{CR}}(\mathcal{A}_1)\,. \tag{3.15}$$

We now turn towards the second difference. We construct $\overline{\mathcal{A}}_{\mathsf{lprf}}$ against $\mathcal{F}'$ as follows. It starts by initialising an empty set $\mathcal{T}$. Upon receiving a query X to F by $\mathcal{A}_{\mathsf{lprf}}$, $\overline{\mathcal{A}}_{\mathsf{lprf}}$ locally computes $H \leftarrow \mathcal{H}(X)$. If $H \in \mathcal{T}$, $\overline{\mathcal{A}}_{\mathsf{lprf}}$ returns $\perp$ to $\mathcal{A}_{\mathsf{lprf}}$. Otherwise, $\overline{\mathcal{A}}_{\mathsf{lprf}}$ adds H to $\mathcal{T}$, sends H to its oracle F to get Y, which it forwards as the response to $\mathcal{A}_{\mathsf{lprf}}$. For leakage queries $(X, L_H, L_{F'})$ to LF, $\overline{\mathcal{A}}_{\mathsf{lprf}}$ first computes $H \leftarrow \mathcal{H}(X)$ and returns $\perp$ if $H \in \mathcal{T}$. If that is not the case, $\overline{\mathcal{A}}_{\mathsf{lprf}}$ locally computes $\Lambda_H \leftarrow L_H(X)$, sends $(H, L_{F'})$ to its own leakage oracle LF to get $(Y, \Lambda_{F'})$, and sends $(Y, (\Lambda_H, \Lambda_{F'}))$ back to $\mathcal{A}_{\mathsf{lprf}}$. When $\mathcal{A}_{\mathsf{lprf}}$ terminates by outputting a bit b', $\overline{\mathcal{A}}_{\mathsf{lprf}}$ outputs b'. It holds that $\overline{\mathcal{A}}_{\mathsf{lprf}}$ perfectly simulates $\boxed{\mathsf{G}}$ for $\mathcal{A}_{\mathsf{lprf}}$ with the same secret bit as its own game LPRF. Hence, we obtain

$$\begin{aligned}\left|\Pr[\mathcal{A}_{\mathsf{lprf}}^{\boxed{\mathsf{G}}} \Rightarrow 1 \mid b = 1] - \Pr[\mathcal{A}_{\mathsf{lprf}}^{\boxed{\mathsf{G}}} \Rightarrow 1 \mid b = 0]\right| &= \Big|\Pr[\overline{\mathcal{A}}_{\mathsf{lprf}}^{\mathsf{LPRF}} \Rightarrow 1 \mid b = 1] \\ &\quad - \Pr[\overline{\mathcal{A}}_{\mathsf{lprf}}^{\mathsf{LPRF}} \Rightarrow 1 \mid b = 0]\Big| \\ &= \mathbf{Adv}_{\mathcal{F}'}^{\mathsf{LPRF}}(\overline{\mathcal{A}}_{\mathsf{lprf}}, \mathcal{L}_{F'})\,. \end{aligned} \tag{3.16}$$

Just as for the first difference, the third one is bound by the probability of $\mathcal{A}$ triggering Bad, except for secret bit fixed to 0, i.e.,

$$\left|\Pr[\mathcal{A}_{\mathsf{lprf}}^{\mathsf{G}} \Rightarrow 1 \mid b = 0] - \Pr[\mathcal{A}_{\mathsf{lprf}}^{\boxed{\mathsf{G}}} \Rightarrow 1 \mid b = 0]\right| \leq \Pr[\mathcal{A}_{\mathsf{lprf}}^{\mathsf{G}} \text{ sets } \mathsf{Bad} \mid b = 0]\,. \tag{3.17}$$

Similar to $\mathcal{A}_1$, we construct $\mathcal{A}_0$ against $\mathcal{H}$. It chooses a key K for $\mathcal{F}'$ at random which will be used for leakage queries by $\mathcal{A}_{\mathsf{lprf}}$. More precisely, for a leakage query $(X, (L_H, L_{F'}))$ to LF, $\mathcal{A}_0$ proceeds as follows. It computes $Y \leftarrow \mathcal{F}'(K, \mathcal{H}(X))$, $\Lambda_H \leftarrow L_H(X)$, and $\Lambda_{F'} \leftarrow L_{F'}(K, H)$ and sends $(Y, (\Lambda_H, \Lambda_{F'}))$ back to $\mathcal{A}_{\mathsf{lprf}}$. For queries X to F by $\mathcal{A}_{\mathsf{lprf}}$, $\mathcal{A}_0$ first computes $H \leftarrow \mathcal{H}(X)$ and $Y \leftarrow \mathcal{F}'(K, H)$, followed by sending Y to $\mathcal{A}_{\mathsf{lprf}}$. Again, $\mathcal{A}_0$ monitors all hash values that occur for the queries made by $\mathcal{A}_{\mathsf{lprf}}$. For the query X that triggers Bad, $\mathcal{A}_0$ looks up X' that forms the collision and outputs (X, X'). We conclude with

$$\Pr[\mathcal{A}_{\mathsf{lprf}}^{\mathsf{G}} \text{ sets } \mathsf{Bad} \mid b = 0] = \mathbf{Adv}_{\mathcal{H}}^{\mathsf{CR}}(\mathcal{A}_0)\,. \tag{3.18}$$

By collecting (3.14), (3.15), (3.17), (3.18), and (3.16), defining $\mathcal{A}_{\mathsf{hash}}$ to be the adversary with the higher advantage among $\mathcal{A}_0$ and $\mathcal{A}_1$ and inserting into (3.13), we obtain

$$\begin{aligned}\mathbf{Adv}_{\hat{\mathcal{F}}}^{\mathsf{LPRF}}(\mathcal{A}_{\mathsf{lprf}}, \mathcal{L}_{\hat{F}}) &\leq \left|\Pr[\mathcal{A}_{\mathsf{lprf}}^{\mathsf{G}} \Rightarrow 1 \mid b = 1] - \Pr[\mathcal{A}_{\mathsf{lprf}}^{\boxed{\mathsf{G}}} \Rightarrow 1 \mid b = 1]\right| \\ &\quad + \left|\Pr[\mathcal{A}_{\mathsf{lprf}}^{\boxed{\mathsf{G}}} \Rightarrow 1 \mid b = 1] - \Pr[\mathcal{A}_{\mathsf{lprf}}^{\boxed{\mathsf{G}}} \Rightarrow 1 \mid b = 0]\right| \\ &\quad + \left|\Pr[\mathcal{A}_{\mathsf{lprf}}^{\boxed{\mathsf{G}}} \Rightarrow 1 \mid b = 0] - \Pr[\mathcal{A}_{\mathsf{lprf}}^{\mathsf{G}} \Rightarrow 1 \mid b = 0]\right| \\ &= \mathbf{Adv}_{\mathcal{H}}^{\mathsf{CR}}(\mathcal{A}_1) + \mathbf{Adv}_{\mathcal{F}'}^{\mathsf{LPRF}}(\overline{\mathcal{A}}_{\mathsf{lprf}}, \mathcal{L}_{F'}) + \mathbf{Adv}_{\mathcal{H}}^{\mathsf{CR}}(\mathcal{A}_0) \\ &\leq 2\,\mathbf{Adv}_{\mathcal{H}}^{\mathsf{CR}}(\mathcal{A}_{\mathsf{hash}}) + \mathbf{Adv}_{\mathcal{F}'}^{\mathsf{LPRF}}(\overline{\mathcal{A}}_{\mathsf{lprf}}, \mathcal{L}_{F'})\,. \end{aligned}$$

This concludes the proof. □

Games $\mathsf{G}, \boxed{\mathsf{G}}$	**oracle** $\mathsf{F}(X)$	**oracle** $\mathsf{LF}(X, (L_H, L_{F'}))$
$K \leftarrow_{\$} \mathcal{K}$	$H \leftarrow \mathcal{H}(X)$	$H \leftarrow \mathcal{H}(X)$
$\mathcal{T} \leftarrow \emptyset$	**if** $b = 0$	$Y \leftarrow \mathcal{F}'(K, H)$
$\mathsf{Bad} \leftarrow \text{false}$	$\quad Y \leftarrow_{\$} \mathcal{Y}$	$\Lambda_H \leftarrow L_H(X)$
$b' \leftarrow \mathcal{A}^{\mathsf{F},\mathsf{LF}}()$	**else**	$\Lambda_{F'} \leftarrow L_{F'}(K, H)$
	$\quad Y \leftarrow \mathcal{F}'(K, H)$	$\Lambda \leftarrow (\Lambda_H, \Lambda_{F'})$
	if $H \in \mathcal{T}$	**if** $H \in \mathcal{T}$
	$\quad \mathsf{Bad} \leftarrow \text{true}$	$\quad \mathsf{Bad} \leftarrow \text{true}$
	$\quad \boxed{\textbf{return } \perp}$	$\quad \boxed{\textbf{return } \perp}$
	$\mathcal{T} \leftarrow_{\cup} H$	$\mathcal{T} \leftarrow_{\cup} H$
	return Y	**return** (Y, Λ)

Figure 3.8: Games used in the proof of Theorem 3.1.6.

3.1.3 The FGHF′ Composition Theorem

Below we state the composition theorem for the FGHF′ construction. It follows from collecting the results above, i.e., Theorem 3.1.1, Corollary 3.1.5, and Theorem 3.1.6, and combining it with the N2 composition theorem by Barwell et al. [BMOS17], i.e., Theorem 2.3.6. The exact implications are visualised in Figure 3.2.

Theorem 3.1.7 (LAE Security of the FGHF′ Construction)**.** *Let $\mathcal{F}$ be a function, $\mathcal{G}$ be a* PRG*, $\mathcal{H}$ be a vector hash function, and $\mathcal{F}'$ be a function with associated leakage sets $\mathcal{L}_F$, $\mathcal{L}_G$, $\mathcal{L}_H$, and $\mathcal{L}_{F'}$, respectively. Let* FGHF′ *be the composition of $\mathcal{F}$, $\mathcal{G}$, $\mathcal{H}$, and $\mathcal{F}'$ with associated leakage sets $\mathcal{L}_{AE} = \mathcal{L}_{VD} = \mathcal{L}_F \times \mathcal{L}_G \times \mathcal{L}_H \times \mathcal{L}_{F'}$. Then, for any* LAE *adversary $\mathcal{A}_{\mathsf{ae}}$ against* FGHF′*, making q queries to* Enc*, there exist adversaries $\mathcal{A}_{\mathsf{lprf}}$, $\overline{\mathcal{A}}_{\mathsf{lprf}}$, $\mathcal{A}_{\mathsf{prg}}$, $\mathcal{A}_{\mathsf{hash}}$, and $\mathcal{A}_{\mathsf{luf}}$ such that*

$$\begin{aligned}\mathbf{Adv}^{\mathsf{LAE}}_{\mathrm{FGHF}'}(\mathcal{A}_{\mathsf{ae}}, \mathcal{L}_{AE}, \mathcal{L}_{VD}) \leq\ & \mathbf{Adv}^{\mathsf{LPRF}}_{\mathcal{F}}(\mathcal{A}_{\mathsf{lprf}}, \mathcal{L}_F) + \mathbf{Adv}^{\mathsf{LPRF}}_{\mathcal{F}'}(\overline{\mathcal{A}}_{\mathsf{lprf}}, \mathcal{L}_{F'}) \\ & + q\,\mathbf{Adv}^{\mathsf{PRG}}_{\mathcal{G}}(\mathcal{A}_{\mathsf{prg}}) + 4\,\mathbf{Adv}^{\mathsf{CR}}_{\mathcal{H}}(\mathcal{A}_{\mathsf{hash}}) \\ & + 2\,\mathbf{Adv}^{\mathsf{LUF}}_{\mathcal{F}'}(\mathcal{A}_{\mathsf{luf}}, \mathcal{L}_{F'})\,.\end{aligned}$$

3.2 Slae: An Instantiation based on Sponges

We introduce Slae, pronounced "sleigh", a **S**ponge-based non-adaptive **L**eakage-resilient **AE**AD scheme. This is achieved by instantiating the FGHF′ construction from Section 3.1 purely from sponges. A salient property is that Slae bears many similarities with a prior sponge-based AEAD scheme called Isap [DEM+17, DEM+20b]. Indeed, Slae is inspired by Isap as is the FGHF′ construction which initially derived from the concrete AEAD scheme Slae.

The AEAD scheme Isap is designed with the goal to inherently resist side-channel attacks by deploying a rekeying mechanism. More specifically, the rekeying mechanism is used to achieve security against Differential Power Analysis (DPA).

Following the N2 construction, ISAP consists of a symmetric encryption scheme, called ISAPENC, and a MAC, called ISAPMAC. Both components are conceived by augmenting well-established sponge constructions using a rekeying function ISAPRK. The DPA resistance stems, intuitively, from two facts: first, ISAP never uses the same key more than once and, second, the slow absorption rate used in the rekeying function. However, the initial proposal of ISAP [DEM+17] was not backed up by any formal security analysis. Note that this is no longer the case for the newer proposal ISAP v2.0, which is accompanied by a formal security analysis [DEM+20b].

The overall structure of ISAP is retained in SLAE but some conceptual changes are done in order to facilitate its security analysis. The design of SLAE can be understood across three different levels of abstraction. At the highest level, just as ISAP, it is the N2 composition of a symmetric encryption scheme SLENC and a MAC SLMAC. At the second abstraction level, SLENC and SLMAC can be viewed in terms of smaller components following the FGHF′ construction. More specifically, SLENC is decomposed into a fixed-input-length function $\mathcal{F}$ and a pseudorandom generator with variable output length $\mathcal{G}$ while SLMAC consists of a vector hash function $\mathcal{H}$ and a fixed-input-length function $\mathcal{F}'$. At this point, note that the generic FGHF′ construction does not use a rekeying approach as opposed to ISAP. At the third abstraction level, SLAE results from instantiating the four components $\mathcal{F}$, $\mathcal{G}$, $\mathcal{H}$, and $\mathcal{F}'$ with the sponge-based primitives SLFUNC, SPRG, and SVHASH. Note that these primitives are based on T-sponges, another difference to ISAP which relies on P-sponges (cf. Section 2.2.7).

The AEAD scheme SLAE in terms of its underlying components SLFUNC, SPRG, and SVHASH, is described in Figure 3.9. The sponge-based instantiations of the components are described in Figure 3.10.

We briefly recall the parameters. Nonces are expected to be of fixed length ν. By α we denote the length of the associated data, while k describes the key length. The output length of the hash function is denoted by w. The length of a message and ciphertext is μ and γ, respectively. Note that these can be arbitrarily long and, in particular, that we use γ to denote both the ciphertext length of the underlying encryption scheme SLENC as well as SLAE. Finally, τ denotes the tag length of the MAC SLMAC.

3.2.1 The SLENC Construction

The encryption scheme SLENC is described in Figure 3.9. It follows the blueprint from the FGHF′ construction, instantiating the encryption scheme $\text{SE}[\mathcal{F}, \mathcal{G}]$ with the sponge-based fixed-input-length function SLFUNC and the pseudorandom generator SPRG. Both SLFUNC and SPRG are described in Figure 3.10, while the former and latter are depicted in Figure 3.11 and Figure 3.12, respectively.

To encrypt a message M, it initialises the initial state of the sponge to $K_e \parallel IV$, where IV is some initialisation vector of length $n - k$. In the absorbing phase (corresponding to SLFUNC), the nonce is absorbed using a reduced rate ζ. After the nonce is fully absorbed, the squeezing phase (corresponding to SPRG) starts. In this phase, every iteration of the sponge construction outputs r-bits until a bit string of length equal to the message is obtained and the ciphertext is obtained by XORing the message to this bit string.

SLAE-Enc(K, N, A, M)

parse K **as** $K_e \parallel K_m$
$C_e \leftarrow$ SLENC-Enc(K_e, N, M)
$T \leftarrow$ SLMAC-Tag(K_m, N, A, C_e)
$C \leftarrow C_e \parallel T$
return C

SLAE-Dec(K, N, A, C)

parse K **as** $K_e \parallel K_m$
parse C **as** $C_e \parallel T$
$V \leftarrow$ SLMAC-Ver(K_m, N, A, C_e, T)
if $V = \bot$
  return $\bot$
$M \leftarrow$ SLENC-Dec(K_e, N, M)
return M

SLENC-Enc(K_e, N, M)

$z \leftarrow$ SLFUNC(K_e, N)
$C_e \leftarrow$ SPRG$(z, |M|) \oplus M$
return C_e

SLMAC-Tag(K_m, N, A, C_e)

$H \leftarrow$ SVHASH(N, A, C_e)
$T \leftarrow$ SLFUNC(K_m, H)
return $\lceil T \rceil^\tau$

SLENC-Dec(K_e, N, C_e)

$z \leftarrow$ SLFUNC(K_e, N)
$M \leftarrow$ SPRG$(z, |C_e|) \oplus C_e$
return M

SLMAC-Ver(K_m, N, A, C_e, T)

$H \leftarrow$ SVHASH(N, A, C_e)
$T' \leftarrow$ SLFUNC(K_m, H)
if $T' \neq T$
  return $\bot$
return $\top$

Figure 3.9: Pseudocode of the AEAD scheme SLAE = (SLAE-Enc, SLAE-Dec) in terms of the sponge-based primitives SLFUNC, SPRG, and SVHASH.

SLFUNC(K, X)

parse X **as** $X_1 \parallel \ldots \parallel X_l$ **s.t.** $\forall i |X| = r$
$Y_0 \leftarrow K \parallel IV$
$S_1 \leftarrow \rho(Y_0)$
for i **in** $\{1, \ldots, l\}$
  $Y_i \leftarrow (\bar{S}_i \oplus X_i) \parallel \hat{S}_i$
  $S_{i+1} \leftarrow \rho(Y_i)$
return S_{l+1}

SPRG(z, L)

$l \leftarrow \lceil \frac{L}{r} \rceil$
$S_1 \leftarrow z$
for i **in** $\{1, \ldots, l-1\}$
  $S_{i+1} \leftarrow \rho(S_i)$
$Z \leftarrow \bar{S}_1 \parallel \ldots \parallel \bar{S}_l$
return $\lceil Z \rceil^L$

$\mathsf{lpad}(X, r)$

$y \leftarrow |X| \mod r$
return $1 \parallel 0^{r-y-1}$

SVHASH(N, A, C)

$\boldsymbol{A} \leftarrow A \parallel \mathsf{lpad}(A, r)$
parse $\boldsymbol{A}$ **as** $\boldsymbol{A}_1 \parallel \ldots \parallel \boldsymbol{A}_u$ **s.t.** $|\boldsymbol{A}_i| = r, \forall i$
$\boldsymbol{C} \leftarrow C \parallel \mathsf{lpad}(C, r)$
parse $\boldsymbol{C}$ **as** $\boldsymbol{C}_1 \parallel \ldots \parallel \boldsymbol{C}_u$ **s.t.** $|\boldsymbol{C}_i| = r, \forall i$
$Y_0 \leftarrow N \parallel IV$
$S_1 \leftarrow \rho(Y_0)$
// Absorb associated data
for i **in** $\{1, \ldots, u\}$
  $Y_i \leftarrow (\bar{S}_i \oplus \boldsymbol{A}_i) \parallel \hat{S}_i$
  $S_{i+1} \leftarrow \rho(Y_i)$
// Input separation
$S_{u+1} \leftarrow \bar{S}_u \parallel (\hat{S}_u \oplus (1 \parallel 0^{c-1}))$
// Absorb ciphertext
for i **in** $\{u+1, \ldots, u+v\}$
  $Y_i \leftarrow (\bar{S}_i \oplus \boldsymbol{C}_i) \parallel \hat{S}_i$
  $S_{i+1} \leftarrow \rho(Y_i)$
$H \leftarrow \lceil S_{u+v+1} \rceil^w$
return H

Figure 3.10: Pseudocode of the sponge-based primitives SLFUNC, SPRG, and SVHASH in terms of a random transformation $\rho\colon \{0,1\}^n \to \{0,1\}^n$, where $n = r + c$.

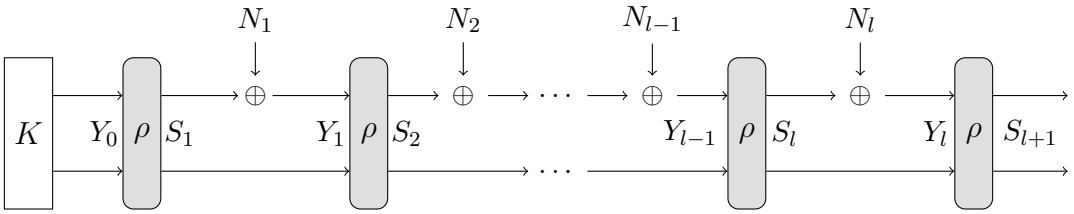

Figure 3.11: Visualisation of the sponge-based function SLFUNC.

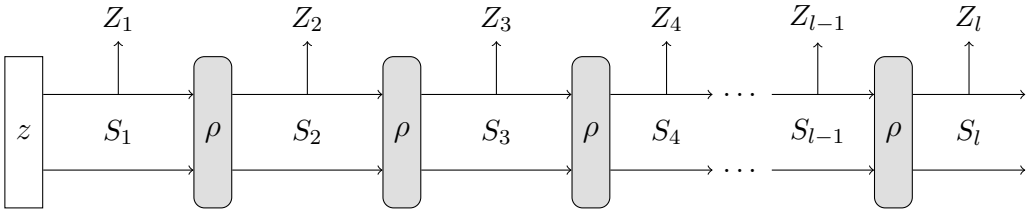

Figure 3.12: Visualisation of the sponge-based pseudorandom generator SPRG.

3.2.2 The SLMAC Construction

The message authentication code SLMAC is described in Figure 3.9. The construction is based on a sponge-based vector hash function SVHASH, which is described in Figure 3.10 and depicted in Figure 3.13. In addition, it uses a fixed-input-length function SLFUNC described in Figure 3.10 and depicted in Figure 3.11.

To authenticate a triple (N, A, C), the sponge state is initialised with $N \parallel IV$ for an initialisation vector IV of length $n - \nu$. Subsequently, the associated data and the ciphertext are padded to length a multiple of the sponge rate r and then absorbed r bits in each iteration. To demarcate the switch from absorbing the associated data to absorbing the ciphertext the bit string $1 \parallel 0^{c-1}$ is XORed to the inner state, when absorbing the first ciphertext block. This ensures that distinct input pairs $(\boldsymbol{A}, \boldsymbol{C}) \neq (\overline{\boldsymbol{A}}, \overline{\boldsymbol{C}})$ with $\boldsymbol{A} \parallel \boldsymbol{C} = \overline{\boldsymbol{A}} \parallel \overline{\boldsymbol{C}}$ still yield different hash values.

Once the hash value is computed, it is fed into SLFUNC, by initialising the sponge state with $K_m \parallel IV$ and absorbing the hash value at a reduced rate ζ. The resulting state is truncated to its leftmost τ bits to obtain the tag.

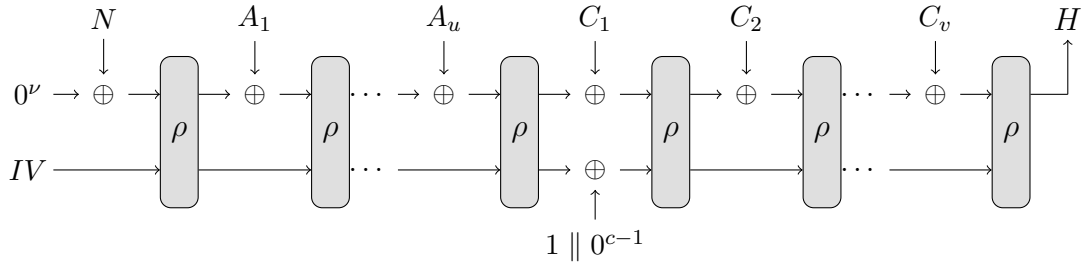

Figure 3.13: Visualisation of the sponge-based hash function SVHASH.

3.2.3 Security of the Sponge-Based Primitives

We now turn towards the security of the sponge-based components underlying SLAE. We prove SLFUNC to be a leakage-resilient function family achieving both pseudorandomness and unpredictability. Informally, we show that by a reduced absorption rate, e.g., $\zeta = 1$, the impact of leakage can be limited to $\lambda 2^{\zeta}$, where λ is the number of bits leaked between consecutive invocations of the transformation ρ. Although sponge-based pseudorandom generators and hash functions are extensively studied [BDPV10, BDPVA07], SPRG and SVHASH are non-standard constructions and as such, their security follows not immediately from existing results. The former uses the seed as the initial state rather than absorbing it, while the latter takes triples of inputs and employs a special separation between these inputs.

Security of SLFUNC

Although LPRF and LUF security are incomparable notions (as we will formally show in Section 3.3.1), it is still possible to meet both notions simultaneously through a single primitive. Indeed, the FGHF′ composition theorem (cf. Theorem 3.1.7) requires that such

a primitive exists since $\mathcal{F}'$ is required to satisfy both security notions. We now show that the SLFUNC construction is well-suited for this role, and in fact that it can be used to instantiate both the $\mathcal{F}$ and $\mathcal{F}'$ components—as is the case in SLAE. Moreover, the most extensively studied leakage-resilient object is that of a pseudorandom function due to its versatility in several potential applications. As a practical construction of this primitive against non-adaptive leakage, SLFUNC is of independent interest. The LPRF security of SLFUNC is stated formally in the following theorem.

Theorem 3.2.1. *Let* SLFUNC *be the function family described in Figure 3.10 taking as input strings of size $(l \cdot \zeta)$ bits and returning τ-bit strings. Then, for any* LPRF *adversary $\mathcal{A}$ against* SLFUNC *and any vector of leakage functions $[L_0, \ldots, L_l]$ where each component maps n bits to λ bits such that $\mathcal{L}_\lambda = \{[L_0, \ldots, L_l]\}$, it holds that*

$$\mathbf{Adv}^{\mathsf{LPRF}}_{\text{SLFUNC}}(\mathcal{A}, \mathcal{L}_\lambda) \leq \frac{q_T(q_T+2) + (q_{\mathsf{F}} + q_{\mathsf{LF}})q_\rho}{2^{n-\zeta-1}} + \frac{2q_\rho}{2^{k-\lambda 2^\zeta}} + \frac{2lq_{\mathsf{F}}q_\rho}{2^{n-\lambda 2^\zeta}}\,.$$

In the above q_ρ, q_{F}, and q_{LF} denote the number of queries $\mathcal{A}$ makes to its oracles ρ, F, and LF, respectively, and $q_T = (l+1)(q_{\mathsf{LF}} + q_{\mathsf{F}}) + q_\rho$. Moreover, it is required that

$$q_\rho + l(q_{\mathsf{F}} + q_{\mathsf{LF}}) \leq 2^{k-1}\,,$$
$$2^\zeta q_\rho + l(q_{\mathsf{F}} + q_{\mathsf{LF}}) \leq 2^{n-1}\,.$$

Proof. We prove the theorem for the case where SLFUNC takes inputs of fixed length $l \cdot \zeta$ and returns outputs of length n. The general case then follows by means of a simple reduction which truncates the output of SLFUNC to the required number of bits. The proof proceeds by considering the following sequence of games.

Game G_0 is displayed in Figure 3.14. It is the game LPRF instantiated with SLFUNC and the secret bit b set to 1, i.e., oracle F always returns the evaluations of SLFUNC. Furthermore, the adversary is also given oracle access to the random transformation ρ, which SLFUNC depends on, where ρ is lazily sampled across all oracles (ρ, F, and LF) using $p[\,]$. Thus,

$$\Pr[\mathcal{A}^{\mathsf{LPRF}} \Rightarrow 1 \,|\, b = 1] = \Pr[\mathcal{A}^{\mathsf{G}_0} \Rightarrow 1]\,. \tag{3.19}$$

Next, we consider game $\boxed{\mathsf{G}_1}$, described in Figure 3.15. The game introduces three flags Bad_1^1, Bad_1^2, and Bad_1^3, which are used to evaluate the last round of the oracle F in a different but equivalent way. Precisely, the transformation used in the last round of F is considered to be special and its values are stored in $p^*[\,]$ instead of $p[\,]$. However, the boxed statements after the three bad events ensure that $p^*[\,]$ and $p[\,]$ are aligned on common inputs. Hence, from point of view of $\mathcal{A}$, G_0 and $\boxed{\mathsf{G}_1}$ are identical, the mere difference being that $\boxed{\mathsf{G}_1}$ uses additionally $p^*[\,]$ internally. This yields

$$\Pr[\mathcal{A}^{\boxed{\mathsf{G}_1}} \Rightarrow 1] = \Pr[\mathcal{A}^{\mathsf{G}_0} \Rightarrow 1]\,. \tag{3.20}$$

We continue with game G_1 displayed in Figure 3.15. It is the same as $\boxed{\mathsf{G}_1}$ without the boxed code, thereby not maintaining the consistency between $p^*[\,]$ and $p[\,]$ on common inputs. Precisely, G_1 corresponds to the game where the last round of F—and hence the

output of F—is evaluated using an independent random transformation. Clearly, G_1 and $\boxed{\mathsf{G}_1}$ are identical until the event $\mathsf{Bad}_1^1 \cup \mathsf{Bad}_1^2 \cup \mathsf{Bad}_1^3$ occurs. Let C and E be events such that $\mathsf{Bad}_1^3 \subseteq C$ and $\mathsf{Bad}_1^1 \subseteq E$—we will define the exact events below. Then it holds that

$$\begin{aligned}\Pr[\mathcal{A}^{\mathsf{G}_1} \Rightarrow 1] - \Pr[\mathcal{A}^{\boxed{\mathsf{G}_1}} \Rightarrow 1] &\leq \Pr[\mathsf{Bad}_1^1 \cup \mathsf{Bad}_1^2 \cup \mathsf{Bad}_1^3] \\ &\leq \Pr[E \cup \mathsf{Bad}_1^2 \cup C] \\ &\leq \Pr[C] + \Pr[E \,|\, \overline{C}] + \Pr[\mathsf{Bad}_1^2 \,|\, \overline{C}, \overline{E}]\,. \end{aligned} \tag{3.21}$$

Let q_ρ, q_{LF}, and q_{F} denote the number of queries $\mathcal{A}$ makes to its oracles ρ, LF, and F, respectively. We can further assume, without loss of generality, that $\mathcal{A}$ never repeats a query to any of its oracles. Furthermore, $\mathcal{A}$ sets Bad_1^3 if it makes two distinct queries, X to F and X' to LF, such that $\textsc{SlFunc}(K, X) = \textsc{SlFunc}(K, X')$.

We observe $\textsc{SlFunc}(K, X)$ can be viewed as the sponge-based hash function $\textsc{SvHash}$ evaluated on input $0 \parallel X$ with an initial state of $K \parallel IV$ (instead of 0^n) and capacity $n - \zeta$. We will use this to bound the probability of event C—and in turn of Bad_1^3—but first define the event C. Let $\mathcal{H}^{K\|IV}$ be the hash function we just described. Then C is the event that one of the following conditions is satisfied.

1. $\mathcal{A}$ makes queries X and X' across F and LF such that:

   $$\mathcal{H}^{K\|IV}(0 \parallel X) = \mathcal{H}^{K\|IV}(0 \parallel X')$$

   and $X \neq X'$. Note that this condition yields $\mathsf{Bad}_1^3 \subseteq C$.

2. $\mathcal{A}$ makes queries X and X' across F and LF such that:

   $$\mathcal{H}^{K\|IV}(0 \parallel X) = \mathcal{H}^{K\|IV}(0 \parallel X'')\,,$$

   for some X'' that is a prefix of X' and $|X''| < |X'|$.

3. $\mathcal{A}$ makes a query X to F or LF such that:

   $$\mathcal{H}^{K\|IV}(0 \parallel X') = K \parallel IV\,,$$

   for some X' that is a prefix of X.

4. $\mathcal{A}$ makes queries X and X' across F and LF such that:

   $$\mathcal{H}^{K\|IV}(0 \parallel X'') = \mathcal{H}^{K\|IV}(0 \parallel X''')\,,$$

   where X'' is a prefix of X, X''' is a prefix of X', $X'' \neq \varepsilon \neq X'''$, and $X'' \neq X'''$.

For any adversary $\mathcal{A}$ that triggers event C, we can construct another adversary $\mathcal{A}_{\mathsf{hash}}$ that finds a collision in the hash function $\mathcal{H}^{K\|IV}$. More concretely, adversary $\mathcal{A}_{\mathsf{hash}}$ knows the key K, which as part of the initial state of the hash function is public. Together with access to ρ, $\mathcal{A}_{\mathsf{hash}}$ can simulate G_1 for $\mathcal{A}$ while constantly checking for the event C. Once C is triggered, $\mathcal{A}_{\mathsf{hash}}$ can directly output a collision. This allows to apply the bound for the sponge-based hash function below. Note that the different initial state is no hindrance

as the bound is independent of it. Thus, applying inequality (3.64) while setting $w = n$ and $c' = n - \zeta$ yields

$$\Pr[\mathsf{Bad}_1^3] \leq \Pr[C] \leq \frac{q_T(q_T+2)}{2^{n-\zeta}}\,, \tag{3.22}$$

where $q_T = (l+1)(q_{\mathsf{LF}} + q_{\mathsf{F}}) + q_\rho$.

We continue with defining event E and bounding the term $\Pr[E \,|\, \overline{C}]$. We use $\mathcal{V}_{\mathsf{F}}$ and $\mathcal{V}_{\mathsf{LF}}$ to denote the set unions of intermediate values $\{Y_1, \ldots, Y_l\}$ evaluated in F and LF, respectively, over all queries that $\mathcal{A}$ has made so far. Let E_j be the event that the j^{th} query Z that $\mathcal{A}$ makes to ρ is such that $Z = Y_0$ or $Z \oplus A \parallel 0^{n-\zeta} \in \mathcal{V}_{\mathsf{F}}$ for some $A \in \{0,1\}^\zeta$ and define $E = E_1 \cup \cdots \cup E_{q_\rho}$. Clearly, we have $\mathsf{Bad}_1^1 \subseteq E$. We further have

$$\begin{aligned}\Pr[E \,|\, \overline{C}] &\leq \Pr[E_1 \cup E_2 \cup \ldots \cup E_{q_\rho} \,|\, \overline{C}] \\ &= \Pr[E_1 \,|\, \overline{C}] + \Pr[E_2 \,|\, \overline{E}_1, \overline{C}] + \ldots + \Pr[E_{q_\rho} \,|\, \overline{E}_1, \ldots, \overline{E}_{q_\rho - 1}, \overline{C}]\,. \end{aligned} \tag{3.23}$$

Conditioned on $\overline{C}$, there are no two queries such that $Y_i = Y_j$ and $i \neq j$. That is, no state $p[Y_i]$ occurs in more than one position, considering queries across both F and LF. In particular, every state is subject to at most one leakage function and no internal state is ever outputted by F or LF. As long as event E does not occur, we further have that the outputs of ρ do not expose the internal states of F and LF. Then, for $i \geq 1$, guessing a value $Y_i \in \mathcal{V}_{\mathsf{F}}$ is tantamount to guessing the corresponding S_i, since $Y_i = S_i \oplus (X_i \parallel 0^{n-\zeta})$ and X_i is known to the adversary. Then by the union bound and the above observations it follows that

$$\begin{aligned}\Pr[E_j \,|\, \overline{E}_1, \ldots, \overline{E}_{j-1}, \overline{C}] &\leq \sum_{Y_i \in \mathcal{V}_{\mathsf{F}}} 2^{-\mathrm{H}_\infty\left(Y_i \,|\, \Lambda, \overline{E}_1, \ldots, \overline{E}_{j-1}, \overline{C}\right)} \\ &\quad + 2^{-\mathrm{H}_\infty\left(K \,|\, \Lambda, \overline{E}_1, \ldots, \overline{E}_{j-1}, \overline{C}\right)} \\ &= \sum_{Y_i \in \mathcal{V}_{\mathsf{F}}} 2^{-\mathrm{H}_\infty\left(S_i \,|\, \Lambda, \overline{E}_1, \ldots, \overline{E}_{j-1}, \overline{C}\right)} \\ &\quad + 2^{-\mathrm{H}_\infty\left(K \,|\, \Lambda, \overline{E}_1, \ldots, \overline{E}_{j-1}, \overline{C}\right)}\,. \end{aligned} \tag{3.24}$$

Even though oracle F does not leak, it can be the case that the same state occurs in a distinct query to LF which leaks. The leakage regarding the key is $L_0(K \parallel IV)$. For a given state S_i, however, the adversary can obtain at most $L_i(S_i \oplus (X_i \parallel 0^{n-\zeta}))$ for each possible value of X_i as leakage. If we consider the aggregate leakage for a given S_i over all possible values of X_i, i.e., $L_i(S_i \oplus 0^\zeta \parallel 0^{n-\zeta}) \parallel \ldots \parallel L_i(S_i \oplus 1^\zeta \parallel 0^{n-\zeta})$, the support of this aggregate random variable is bounded by $2^{\lambda 2^\zeta}$. Then by applying Lemma 2.3.8 we have that

$$\begin{aligned}\mathrm{H}_\infty\left(K \;\middle|\; \Lambda_0, \overline{E}_1, \ldots, \overline{E}_{j-1}, \overline{C}\right) &\geq \mathrm{H}_\infty\left(K \;\middle|\; \overline{E}_1, \ldots, \overline{E}_{j-1}, \overline{C}\right) - \lambda 2^\zeta \\ \mathrm{H}_\infty\left(S_i \;\middle|\; \Lambda_i, \overline{E}_1, \ldots, \overline{E}_{j-1}, \overline{C}\right) &\geq \mathrm{H}_\infty\left(S_i \;\middle|\; \overline{E}_1, \ldots, \overline{E}_{j-1}, \overline{C}\right) - \lambda 2^\zeta\,. \end{aligned} \tag{3.25}$$

It remains to bound the min-entropies of K and S_i conditioned on $E_1, \ldots, E_{j-1}$ and C not occurring. Conditioning on $\overline{E}_j$ excludes one value for the variable K and 2^ζ potential

values for S_i. Conditioning on $\overline{C}$ excludes at most $(l+1)(q_{\mathsf{F}}+q_{\mathsf{LF}})$ from the possible values for K and S_i. By constraining the adversary's queries such that $q_\rho+(l+1)(q_{\mathsf{F}}+q_{\mathsf{LF}}) \leq 2^{k-1}$ and $(2^\zeta)q_\rho + (l+1)(q_{\mathsf{F}}+q_{\mathsf{LF}}) \leq 2^{n-1}$, we obtain, for all possible j, the following bounds

$$\begin{aligned} \mathrm{H}_\infty\left(K \mid \overline{E}_1, \ldots, \overline{E}_{j-1}, \overline{C}\right) &\geq k-1 \\ \mathrm{H}_\infty\left(S_i \mid \overline{E}_1, \ldots, \overline{E}_{j-1}, \overline{C}\right) &\geq n-1\,. \end{aligned} \tag{3.26}$$

Combining (3.24), (3.25), and (3.26) and applying the bound $|\mathcal{V}_{\mathsf{F}}| \leq lq_{\mathsf{F}}$ yields

$$\Pr[E_j \mid \overline{E}_1, \ldots, \overline{E}_{j-1}, \overline{C}] \leq \frac{1}{2^{k-\lambda 2^\zeta - 1}} + \frac{lq_{\mathsf{F}}}{2^{n-\lambda 2^\zeta - 1}}\,. \tag{3.27}$$

We then substitute (3.27) in (3.23) to obtain

$$\Pr[\mathsf{Bad}_1^1 \mid \overline{C}] \leq \Pr[E \mid \overline{C}] \leq \frac{q_\rho}{2^{k-\lambda 2^\zeta - 1}} + \frac{lq_{\mathsf{F}}q_\rho}{2^{n-\lambda 2^\zeta - 1}}\,. \tag{3.28}$$

Finally, we bound the last term of inequality (3.21), that is, $\Pr[\mathsf{Bad}_1^2 \mid \overline{C}, \overline{E}]$. This flag is set if some query by $\mathcal{A}$ to F results in Y_l already contained in $\mathsf{inset}(p)$. If $Y_l \in \mathcal{V}_{\mathsf{F}} \cup \mathcal{V}_{\mathsf{LF}}$ when Bad_1^2 occurs, then C occurred as well. Hence, when conditioning on $\overline{C}$, Bad_1^2 can only be set if $\mathcal{A}$ already queried Y_l to ρ. Let X^* represent $\mathcal{A}$'s query to F that sets Bad_1^2, resulting in the corresponding values $Y_i^* = S_i^* \oplus (X_i^* \parallel 0^{n-\zeta})$ for $1 \leq i \leq l$, such that Y_l^* was already queried to ρ. Conditioned on E not occurring, the variable S_l^* cannot be sampled during a query to ρ. Hence, $p[Y_{l-1}^*]$ must have been sampled during some query to either F or LF after $\mathcal{A}$ has queried Y_l^* to ρ. Now let F be the event that $\mathcal{A}$ makes a query to F or LF resulting in S_l^* such that for some $A \in \{0,1\}^\zeta$ the value $S_l^* \oplus (A \parallel 0^{n-\zeta})$ was already queried to ρ. It then follows that

$$\Pr[\mathsf{Bad}_1^2 \mid \overline{C}, \overline{E}] \leq \Pr[F \mid \overline{C}, \overline{E}]\,. \tag{3.29}$$

We now bound $\Pr[F \mid \overline{C}, \overline{E}]$. Let F_j denote the event that F occurs at the j^{th} query that $\mathcal{A}$ makes to F or LF. Then, by the union bound we have that

$$\Pr[F \mid \overline{C}, \overline{E}] \leq \Pr[F_1 \mid \overline{C}, \overline{E}] + \Pr[F_2 \mid \overline{C}, \overline{E}] + \ldots + \Pr[F_{q_{\mathsf{F}}+q_{\mathsf{LF}}} \mid \overline{C}, \overline{E}]\,. \tag{3.30}$$

Then for each query there are at most $(2^\zeta)q_\rho$ values that S_l^* can take to set F out of $2^n-(l+1)(q_{\mathsf{F}}+q_{\mathsf{LF}})-(2^\zeta)q_\rho$ (due to conditioning on $\overline{C}$ and $\overline{E}$). Thus, by combining (3.29) and (3.30), we get

$$\begin{aligned} \Pr[F \mid \overline{C}, \overline{E}] &\leq \sum_{j=1}^{q_{\mathsf{F}}+q_{\mathsf{LF}}} \frac{(2^\zeta)q_\rho}{2^n - (l+1)(q_{\mathsf{F}}+q_{\mathsf{LF}}) - (2^\zeta)q_\rho} \\ &\leq \frac{2^\zeta(q_{\mathsf{F}}+q_{\mathsf{LF}})q_\rho}{2^{n-1}} \\ &= \frac{(q_{\mathsf{F}}+q_{\mathsf{LF}})q_\rho}{2^{n-\zeta-1}}\,. \end{aligned} \tag{3.31}$$

Finally, combining inequalities (3.21), (3.22), (3.28), and (3.31) we obtain

$$\Pr[\mathcal{A}^{\mathsf{G}_1} \Rightarrow 1] - \Pr[\mathcal{A}^{\boxed{\mathsf{G}_1}} \Rightarrow 1] \leq \frac{q_T(q_T+2)}{2^{n-\zeta}} + \frac{q_\rho}{2^{k-\lambda 2^\zeta - 1}} + \frac{l q_{\mathsf{F}} q_\rho}{2^{n-\lambda 2^\zeta - 1}} + \frac{(q_{\mathsf{F}} + q_{\mathsf{LF}}) q_\rho}{2^{n-\zeta-1}}\,, \tag{3.32}$$

where $q_T = (l+1)(q_{\mathsf{LF}} + q_{\mathsf{F}}) + q_\rho$.

Now we introduce game $\boxed{\mathsf{G}_2}$ which is described in Figure 3.16. Here, the flags Bad_1^1, Bad_1^2, and Bad_1^3 are removed and a new flag Bad_2 is introduced. Notably, in tandem with $p^*[\,]$ we maintain an additional array $f[\,]$, where the latter is indexed by strings of size l instead of n. Then in every evaluation of F we sample a random string, store it in $f[X]$, and then copy this value to $p^*[Y_l]$. However, if $p^*[Y_l]$ was already defined, then Bad_2 is set and the random string in $f[X]$ is replaced with $p^*[Y_l]$. This ensures that no entry in $p^*[\,]$ is ever overwritten. It follows that $\boxed{\mathsf{G}_2}$ is equivalent to G_1 from point of view of $\mathcal{A}$ as they only differ in the internal bookkeeping of variables, hence

$$\Pr[\mathcal{A}^{\boxed{\mathsf{G}_2}} \Rightarrow 1] = \Pr[\mathcal{A}^{\mathsf{G}_1} \Rightarrow 1]\,. \tag{3.33}$$

Game G_2 is the same as $\boxed{\mathsf{G}_2}$ minus the boxed code. Without keeping consistency between $f[\,]$ and $p^*[\,]$, oracle F behaves exactly like game LPRF with secret bit $b = 0$ which leads to

$$\Pr[\mathcal{A}^{\mathsf{LPRF}} \Rightarrow 1 \,|\, b = 0] = \Pr[\mathcal{A}^{\mathsf{G}_2} \Rightarrow 1]\,. \tag{3.34}$$

It is easy to see that G_2 and $\boxed{\mathsf{G}_2}$ can only be distinguished if Bad_2 occurs. This happens if $\mathcal{A}$ makes two queries X and X' to F that result in the same state Y_l. Just as for game G_1, this can be viewed as a collision on $\mathcal{H}^{K\|IV}$ with l rounds. By applying inequality (3.64), we have

$$\Pr[\mathcal{A}^{\mathsf{G}_2} \Rightarrow 1] - \Pr[\mathcal{A}^{\boxed{\mathsf{G}_2}} \Rightarrow 1] \leq \Pr[\mathsf{Bad}_2] \leq \frac{q'_T(q'_T+2)}{2^{n-\zeta}}\,, \tag{3.35}$$

where $q'_T = l(q_{\mathsf{LF}} + q_{\mathsf{F}}) + q_\rho$. Combining inequalities (3.19), (3.20), (3.32), (3.33), (3.35), and (3.34) yields the desired result. □

The next theorem shows that SLFUNC is a good LUF. Its proof bears some similarity to that of Theorem 3.2.1 as it uses similar ideas. However, one important difference lies in the leakage model that is used in this theorem. Since the Lkg oracle returns only the leakage and no output, we add here an extra leakage function that returns the leakage on the output of SLFUNC. In the LPRF case this was not required since in that game the leakage oracle returns the full output anyway.

Theorem 3.2.2. *Let* SLFUNC *be the function family described in Figure 3.10 taking as input strings of size $(l \cdot \zeta)$ bits and returning τ-bit long strings. Then for any* LUF *adversary $\mathcal{A}$ against* SLFUNC*, and any vector of leakage functions $[L_0, \ldots, L_{l+1}]$ where each component maps n bits to λ bits and letting $\mathcal{L}_\lambda = \{[L_0, \ldots, L_{l+1}]\}$, it holds that*

$$\mathbf{Adv}^{\mathsf{LUF}}_{\text{SLFUNC}}(\mathcal{A}, \mathcal{L}_\lambda) \leq \frac{q_T(q_T+2)}{2^{n-\zeta}} + \frac{q_\rho}{2^{k-\lambda 2^\zeta - 1}} + \frac{l q_{\mathsf{Lkg}} q_\rho}{2^{n-\lambda 2^\zeta - 1}} + \frac{q_{\mathsf{Guess}}}{2^{\tau-\lambda-1}}\,.$$

Game G_0

$b \leftarrow_\$ \{0,1\}$
$K \leftarrow_\$ \mathcal{K}$
$b' \leftarrow \mathcal{A}^{\mathsf{F},\mathsf{LF},\rho}()$

oracle $\rho(Z)$

if $p[Z] = \bot$
  $p[Z] \leftarrow_\$ \{0,1\}^n$
return $p[Z]$

oracle $\mathsf{F}(X)$

parse X **as** $X_1 \| \ldots \| X_l$
  s.t. $\forall i\ |X_i| = \zeta$
$Y_0 \leftarrow K \| IV$
if $p[Y_0] = \bot$
  $p[Y_0] \leftarrow_\$ \{0,1\}^n$
$S_1 \leftarrow p[Y_0]$
for i **in** $\{1, \ldots, l\}$
  $Y_i \leftarrow (\bar{S}_i \oplus X_i) \| \hat{S}_i$
  if $p[Y_i] = \bot$
    $p[Y_i] \leftarrow_\$ \{0,1\}^n$
  $S_{i+1} \leftarrow p[Y_i]$
return S_{l+1}

oracle $\mathsf{LF}(X, L_0, \ldots, L_l)$

parse X **as** $X_1 \| \ldots \| X_l$
  s.t. $\forall i\ |X_i| = \zeta$
$Y_0 \leftarrow K \| IV$
$\Lambda \leftarrow L_0(Y_0)$
if $p[Y_0] = \bot$
  $p[Y_0] \leftarrow_\$ \{0,1\}^n$
$S_1 \leftarrow p[Y_0]$
for i **in** $\{1, \ldots, l\}$
  $Y_i \leftarrow (\bar{S}_i \oplus X_i) \| \hat{S}_i$
  $\Lambda \leftarrow \Lambda \| L_i(Y_i)$
  if $p[Y_i] = \bot$
    $p[Y_i] \leftarrow_\$ \{0,1\}^n$
  $S_{i+1} \leftarrow p[Y_i]$
return (S_{l+1}, Λ)

Figure 3.14: Game G_0 used in the proof of Theorem 3.2.1.

Games $\mathsf{G}_1, \boxed{\mathsf{G}_1}$

$b \leftarrow_\$ \{0,1\}$
$K \leftarrow_\$ \mathcal{K}$
$b' \leftarrow \mathcal{A}^{\mathsf{F},\mathsf{LF},\rho}()$

oracle $\rho(Z)$

if $p[Z] = \bot$
  $p[Z] \leftarrow_\$ \{0,1\}^n$
if $Z \in \mathsf{inset}(p^*)$
  $\mathsf{Bad}_1^1 \leftarrow \text{true}$
  $\boxed{p[Z] \leftarrow p^*[Z]}$
return $p[Z]$

oracle $\mathsf{F}(X)$

parse X **as** $X_1 \| \ldots \| X_l$
  s.t. $\forall i\ |X_i| = \zeta$
$Y_0 \leftarrow K \| IV$
if $p[Y_0] = \bot$
  $p[Y_0] \leftarrow_\$ \{0,1\}^n$
$S_1 \leftarrow p[Y_0]$
for i **in** $\{1, \ldots, l-1\}$
  $Y_i \leftarrow (\bar{S}_i \oplus X_i) \| \hat{S}_i$
  if $p[Y_i] = \bot$
    $p[Y_i] \leftarrow_\$ \{0,1\}^n$
  $S_{i+1} \leftarrow p[Y_i]$
$Y_l \leftarrow (\bar{S}_l \oplus X_l) \| \hat{S}_l$
if $p^*[Y_l] = \bot$
  $p^*[Y_l] \leftarrow_\$ \{0,1\}^n$
if $Y_l \in \mathsf{inset}(p)$
  $\mathsf{Bad}_1^2 \leftarrow \text{true}$
  $\boxed{p^*[Y_l] \leftarrow p[Y_l]}$
$S_{l+1} \leftarrow p^*[Y_l]$
return S_{l+1}

oracle $\mathsf{LF}(X, L_0, \ldots, L_l)$

parse X **as** $X_1 \| \ldots \| X_l$
  s.t. $\forall i\ |X_i| = \zeta$
$Y_0 \leftarrow K \| IV$
$\Lambda \leftarrow L_0(Y_0)$
if $S_1 = \bot$
  $S_1 \leftarrow_\$ \{0,1\}^n$
for i **in** $\{1, \ldots, l\}$
  $Y_i \leftarrow (\bar{S}_i \oplus X_i) \| \hat{S}_i$
  $\Lambda \leftarrow \Lambda \| L_i(Y_i)$
  if $p[Y_i] = \bot$
    $p[Y_i] \leftarrow_\$ \{0,1\}^n$
  if $i = l \wedge Y_i \in \mathsf{inset}(p^*)$
    $\mathsf{Bad}_1^3 \leftarrow \text{true}$
    $\boxed{p[Y_l] \leftarrow p^*[Y_l]}$
  $S_{i+1} \leftarrow p[Y_i]$
return (S_{l+1}, Λ)

Figure 3.15: Games G_1 and $\boxed{\mathsf{G}_1}$ used in the proof of Theorem 3.2.1.

Games $\mathsf{G}_2, \boxed{\mathsf{G}_2}$

$b \leftarrow\!\!\$\ \{0,1\}$
$K \leftarrow\!\!\$\ \mathcal{K}$
$b' \leftarrow \mathcal{A}^{\mathsf{F},\mathsf{LF},\rho}()$

oracle $\rho(Z)$

if $p[Z] = \bot$
  $p[Z] \leftarrow\!\!\$\ \{0,1\}^n$
return $p[Z]$

oracle $\mathsf{F}(X)$

parse X **as** $X_1 \parallel \ldots \parallel X_l$
  s.t. $\forall i\ |X_i| = \zeta$
$Y_0 \leftarrow K \parallel IV$
if $p[Y_0] = \bot$
  $p[Y_0] \leftarrow\!\!\$\ \{0,1\}^n$
$S_1 \leftarrow p[Y_0]$
for i **in** $\{1, \ldots, l-1\}$
  $Y_i \leftarrow (\bar{S}_i \oplus X_i) \parallel \hat{S}_i$
  if $p[Y_i] = \bot$
    $p[Y_i] \leftarrow\!\!\$\ \{0,1\}^n$
  $S_{i+1} \leftarrow p[Y_i]$
$Y_l \leftarrow (\bar{S}_l \oplus X_l) \parallel \hat{S}_l$
$f[X] \leftarrow\!\!\$\ \{0,1\}^n$
if $p^*[Y_l] \neq \bot$
  $\mathsf{Bad}_2 \leftarrow$ true
  $\boxed{f[X] \leftarrow\!\!\$\ p^*[Y_l]}$
$p^*[Y_l] \leftarrow\!\!\$\ f[X]$
$S_{l+1} \leftarrow p^*[Y_l]$
return S_{l+1}

oracle $\mathsf{LF}(X, L_0, \ldots, L_l)$

parse X **as** $X_1 \parallel \ldots \parallel X_l$
  s.t. $\forall i\ |X_i| = \zeta$
$Y_0 \leftarrow K \parallel IV$
$\Lambda \leftarrow L_0(Y_0)$
if $S_1 = \bot$
  $S_1 \leftarrow\!\!\$\ \{0,1\}^n$
for i **in** $\{1, \ldots, l\}$
  $Y_i \leftarrow (\bar{S}_i \oplus X_i) \parallel \hat{S}_i$
  $\Lambda \leftarrow \Lambda \parallel L_i(Y_i)$
  if $p[Y_i] = \bot$
    $p[Y_i] \leftarrow\!\!\$\ \{0,1\}^n$
  $S_{i+1} \leftarrow p[Y_i]$
return (S_{l+1}, Λ)

Figure 3.16: Games G_2 and $\boxed{\mathsf{G}_2}$ used in the proof of Theorem 3.2.1.

In the above q_ρ, q_{F}, q_{Lkg} *and* q_{Guess} *denote the number of queries* $\mathcal{A}$ *makes to its oracles* ρ, F, Lkg, *and* Guess, *respectively, and* $q_T = (l+1)(q_{\mathsf{F}} + q_{\mathsf{Lkg}} + q_{\mathsf{Guess}}) + q_\rho$. *Moreover, it is required that the following conditions be satisfied*

$$\begin{aligned} q_\rho + (l+1)(q_{\mathsf{F}} + q_{\mathsf{Lkg}} + q_{\mathsf{Guess}}) &\leq 2^{k-1}\,, \\ 2^\zeta q_\rho + (l+1)(q_{\mathsf{F}} + q_{\mathsf{Lkg}} + q_{\mathsf{Guess}}) &\leq 2^{n-1}\,, \\ q_{\mathsf{Guess}} + (l+1)(q_{\mathsf{F}} + q_{\mathsf{Lkg}} + q_{\mathsf{Guess}}) &\leq 2^{n-1}\,. \end{aligned}$$

Proof. We assume that SLFUNC takes inputs of fixed length $(l \cdot \zeta)$ and returns outputs of length τ. Consider the game described in Figure 3.17. It is the game LUF instantiated with SLFUNC where the adversary is also given oracle access to the random transformation ρ, which SLFUNC depends on. This random transformation is sampled lazily across all oracles and the corresponding values are stored in a global array $p[\,]$. Thus,

$$\Pr[\mathsf{LUF}^{\mathcal{A}} \Rightarrow 1] = \Pr[\mathsf{G}^{\mathcal{A}} \Rightarrow 1]\,. \tag{3.36}$$

By $\mathcal{V}_{\mathsf{Lkg}}$ we denote, at any point in time, the set of intermediate values $\{Y_1, \ldots, Y_l\}$ evaluated in Lkg up to that point, i.e., the states for which the adversary obtained leakage. Similarly, we denote the set of values S_{l+1} evaluated in Lkg by $\mathcal{U}_{\mathsf{Lkg}}$, i.e., the output values for which $\mathcal{A}$ only obtains the leakage. We further define W and B to be the event that $\mathcal{A}$ wins and makes a query Z to ρ such that $Z \in \mathcal{V}_{\mathsf{Lkg}} \cup \{Y_0\}$, respectively. For any event C it holds that

$$\begin{aligned} \Pr[\mathsf{G}^{\mathcal{A}} \Rightarrow 1] &= \Pr[W] \\ &\leq \Pr[W \cup B \cup C] \\ &\leq \Pr[C] + \Pr[B \,|\, \overline{C}] + \Pr[W \,|\, \overline{C}, \overline{B}]\,. \end{aligned} \tag{3.37}$$

We observe $\textsc{SlFunc}(K, X)$ can be viewed as the sponge-based hash function SVHASH evaluated on input $0 \parallel X$ with an initial state of $K \parallel IV$ (instead of 0^n) and capacity $n - \zeta$. We will use this to bound the probability of event C but first define the event C. Let $\mathcal{H}^{K\|IV}$ be the hash function we described above and C be the event that one of the following conditions is satisfied.

1. $\mathcal{A}$ makes queries X and X' across F, Lkg, and Guess such that:

   $$\mathcal{H}^{K\|IV}(0 \parallel X) = \mathcal{H}^{K\|IV}(0 \parallel X')$$

   and $X \neq X'$.

2. $\mathcal{A}$ makes queries X and X' across F, Lkg, and Guess such that:

   $$\mathcal{H}^{K\|IV}(0 \parallel X) = \mathcal{H}^{K\|IV}(0 \parallel X'')$$

   for some X'' that is a prefix of X' and $|X''| < |X'|$.

3. $\mathcal{A}$ makes a query X to F, Lkg, or Guess such that:

   $$\mathcal{H}^{K\|IV}(0 \parallel X') = K \parallel IV$$

   for some X' that is a prefix of X.

4. $\mathcal{A}$ makes queries X and X' across F, Lkg, and Guess such that:

$$\mathcal{H}^{K\|IV}(0 \parallel X'') = \mathcal{H}^{K\|IV}(0 \parallel X''')\,,$$

where X'' is a prefix of X, X''' is a prefix of X', $X'' \neq \varepsilon \neq X'''$, and $X'' \neq X'''$.

For any adversary $\mathcal{A}$ that triggers event C, we can construct another adversary $\mathcal{A}_{\mathsf{hash}}$ that finds a collision in the hash function $\mathcal{H}^{K\|IV}$. More concretely, adversary $\mathcal{A}_{\mathsf{hash}}$ knows the key K, which as part of the initial state of the hash function is public. Together with access to ρ, $\mathcal{A}_{\mathsf{hash}}$ can simulate G_1 for $\mathcal{A}$ while constantly checking for the event C. Once C is triggered, $\mathcal{A}_{\mathsf{hash}}$ can directly output a collision. This allows to apply the bound for the sponge-based hash function below. Note that the different initial state is no hindrance as the bound is independent of it. Thus, applying inequality (3.64) while setting $w = n$ and $c' = n - \zeta$ yields

$$\Pr[C] \leq \frac{q_T(q_T+2)}{2^{n-\zeta}}\,, \tag{3.38}$$

where $q_T = (l+1)(q_{\mathsf{F}} + q_{\mathsf{Lkg}} + q_{\mathsf{Guess}}) + q_\rho$.

We now turn towards bounding the term $\Pr[B \,|\, \overline{C}]$. Let B_j denote the event that B occurs during the j^{th} query to ρ by $\mathcal{A}$. Then it follows that

$$\begin{aligned}\Pr[B \,|\, \overline{C}] &\leq \Pr[B_1 \cup B_2 \cup \cdots \cup B_{q_\rho} \,|\, \overline{C}] \\ &= \Pr[B_1 \,|\, \overline{C}] + \Pr[B_2 \,|\, \overline{B}_1, \overline{C}] + \cdots + \Pr[B_{q_\rho} \,|\, \overline{B}_1, \ldots, \overline{B}_{q_\rho - 1}, \overline{C}]\,.\end{aligned} \tag{3.39}$$

Conditioned on $\overline{C}$, there are no two queries such that $Y_i = Y_j$ and $i \neq j$. That is, no state $p[Y_i]$ occurs in more than one position, considering queries across both F, Lkg, and Guess. In particular, every state is subject to at most one leakage function and no internal state is ever outputted by F. Moreover, as long as B does not occur, the outputs of ρ will never expose any of the state variables contained in $\mathcal{V}_{\mathsf{Lkg}} \cup \mathcal{U}_{\mathsf{Lkg}} \cup \{Y_0\}$.

Then, for $i \geq 1$, guessing a value $Y_i \in \mathcal{V}_{\mathsf{Lkg}}$ is tantamount to guessing the corresponding S_i, since $Y_i = S_i \oplus (X_i \parallel 0^{n-\zeta})$ and X_i is known to the adversary. Then by the union bound and the above observations, for all $j \in [q_\rho]$, it follows that

$$\begin{aligned}\Pr[B_j \,|\, \overline{B}_1, \ldots, \overline{B}_{j-1}, \overline{C}] &\leq \sum_{Y_i \in \mathcal{V}_{\mathsf{Lkg}}} 2^{-\mathrm{H}_\infty\left(Y_i \,\middle|\, \Lambda, \overline{B}_1, \ldots, \overline{B}_{j-1}, \overline{C}\right)} \\ &\quad + 2^{-\mathrm{H}_\infty\left(K \,\middle|\, \Lambda, \overline{B}_1, \ldots, \overline{B}_{j-1}, \overline{C}\right)} \\ &= \sum_{Y_i \in \mathcal{V}_{\mathsf{Lkg}}} 2^{-\mathrm{H}_\infty\left(S_i \,\middle|\, \Lambda, \overline{B}_1, \ldots, \overline{B}_{j-1}, \overline{C}\right)} \\ &\quad + 2^{-\mathrm{H}_\infty\left(K \,\middle|\, \Lambda, \overline{B}_1, \ldots, \overline{B}_{j-1}, \overline{C}\right)}\,.\end{aligned} \tag{3.40}$$

With respect to the key K, the leakage only provides $L_0(K \parallel IV)$ to the adversary. For a given state S_i, however, the adversary may obtain at most $L_i(S_i \oplus X_i \parallel 0^{n-\zeta})$ for each possible value of X_i as leakage. If we consider the aggregate leakage for a given S_i over all possible values of X_i, i.e., $L_i(S_i \oplus 0^\zeta \parallel 0^{n-\zeta}) \parallel \ldots \parallel L_i(S_i \oplus 1^\zeta \parallel 0^{n-\zeta})$, the support of this

aggregate random variable is bounded by $2^{\lambda 2^{\zeta}}$. Then by applying Lemma 2.3.8 we have that

$$\begin{aligned}\mathrm{H}_{\infty}\left(K \mid \Lambda_0, \overline{B}_1, \ldots, \overline{B}_{j-1}, \overline{C}\right) &\geq \mathrm{H}_{\infty}\left(K \mid \overline{B}_1, \ldots, \overline{B}_{j-1}, \overline{C}\right) - \lambda 2^{\zeta} \\ \mathrm{H}_{\infty}\left(S_i \mid \Lambda_i, \overline{B}_1, \ldots, \overline{B}_{j-1}, \overline{C}\right) &\geq \mathrm{H}_{\infty}\left(S_i \mid \overline{B}_1, \ldots, \overline{B}_{j-1}, \overline{C}\right) - \lambda 2^{\zeta}\,. \end{aligned} \tag{3.41}$$

It remains to bound the min-entropies of K and S_i conditioned on $\overline{B}_1, \ldots, \overline{B}_{j-1}$. Conditioning of $\overline{B}_j$ excludes one value for the variable K and 2^{ζ} possible values for the variable S_i. This is because for any query Z that the adversary makes to ρ, the values $(Z \oplus 0^{\zeta} \parallel 0^{n-\zeta}), \ldots, (Z \oplus 1^{\zeta} \parallel 0^{n-\zeta})$ may all be contained in $\mathcal{V}_{\mathsf{Lkg}}$. Conditioning of $\overline{C}$ excludes at most a further $(l+1)(q_{\mathsf{F}}+q_{\mathsf{Lkg}}+q_{\mathsf{Guess}})$ from the possible values that K and S_i may take. By constraining the adversary's queries such that $q_{\rho}+(l+1)(q_{\mathsf{F}}+q_{\mathsf{Lkg}}+q_{\mathsf{Guess}}) \leq 2^{k-1}$ and $(2^{\zeta})q_{\rho} + (l+1)(q_{\mathsf{F}} + q_{\mathsf{Lkg}} + q_{\mathsf{Guess}}) \leq 2^{n-1}$, we obtain, for all possible j, the following bounds

$$\begin{aligned}\mathrm{H}_{\infty}\left(K \mid \overline{B}_1, \ldots, \overline{B}_{j-1}, \overline{C}\right) &\geq k-1 \\ \mathrm{H}_{\infty}\left(S_i \mid \overline{B}_1, \ldots, \overline{B}_{j-1}, \overline{C}\right) &\geq n-1\,. \end{aligned} \tag{3.42}$$

Combining (3.40), (3.41), and (3.42) and applying the bound $|\mathcal{V}_{\mathsf{Lkg}}| \leq l q_{\mathsf{Lkg}}$ yields

$$\Pr[B_j \mid \overline{B}_1, \ldots, \overline{B}_{j-1}, \overline{C}] \leq \frac{1}{2^{k-\lambda 2^{\zeta}-1}} + \frac{l q_{\mathsf{Lkg}}}{2^{n-\lambda 2^{\zeta}-1}}\,. \tag{3.43}$$

We then substitute (3.43) in (3.39) to obtain

$$\Pr[B \mid \overline{C}] \leq \frac{q_{\rho}}{2^{k-\lambda 2^{\zeta}-1}} + \frac{l q_{\mathsf{Lkg}} q_{\rho}}{2^{n-\lambda 2^{\zeta}-1}}\,. \tag{3.44}$$

Finally, we bound the event W conditioned on $\overline{C}$ and $\overline{B}$. Let W_j denote the event that W occurs on the j^{th} query that $\mathcal{A}$ makes to Guess. Then we obtain

$$\begin{aligned}\Pr[W \mid \overline{C}, \overline{B}] &\leq \Pr[W_1 \cup W_2 \cup \cdots \cup W_{q_{\mathsf{Guess}}} \mid \overline{C}, \overline{B}] \\ &= \Pr[W_1 \mid \overline{C}, \overline{B}] + \Pr[W_2 \mid \overline{W}_1, \overline{C}, \overline{B}] + \cdots \\ &\quad + \Pr[W_{q_{\mathsf{Guess}}} \mid \overline{W}_1, \ldots, \overline{W}_{q_{\mathsf{Guess}}-1}, \overline{C}, \overline{B}]\,. \end{aligned} \tag{3.45}$$

By conditioning on $\overline{C}$ and $\overline{B}$, it must be that if W occurs the value of S_{l+1} corresponding to that query must be either new, i.e., freshly sampled, or contained in $\mathcal{U}_{\mathsf{Lkg}}$. In either case, for all $j \in [q_{\mathsf{Guess}}]$, we have that

$$\Pr[W_j \mid \overline{W}_1, \ldots, \overline{W}_{j-1}, \overline{C}, \overline{B}] \leq 2^{-\mathrm{H}_{\infty}\left(\lceil S_{l+1} \rceil^{\tau} \mid \Lambda, \overline{W}_1, \ldots, \overline{W}_{j-1}, \overline{C}, \overline{B}\right)}\,, \tag{3.46}$$

where the case that S_{l+1} is new corresponds to $\Lambda = \varepsilon$. Thus, since Λ is at most $L_{l+1}(S_{l+1})$, by Lemma 2.3.8 we have that

$$\mathrm{H}_{\infty}\left(S_{l+1} \mid \Lambda, \overline{W}_1, \ldots, \overline{W}_{j-1}, \overline{C}, \overline{B}\right) \geq \mathrm{H}_{\infty}\left(S_{l+1} \mid \overline{W}_1, \ldots, \overline{W}_{j-1}, \overline{C}, \overline{B}\right) - \lambda\,. \tag{3.47}$$

Once again conditioning on $\overline{C}$ excludes at most $(l+1)(q_{\mathsf{F}} + q_{\mathsf{Lkg}} + q_{\mathsf{Guess}})$ from the possible values that S_{l+1} may take, whereas conditioning on $\overline{B}$ does not affect S_{l+1}. Moreover,

for each event W_j, conditioning on it not occurring excludes one value from the pool of values that S_{l+1} may take. To cater for this, we require that the adversary's queries satisfy $q_{\mathsf{Guess}} + (l+1)(q_{\mathsf{F}} + q_{\mathsf{Lkg}} + q_{\mathsf{Guess}}) \leq 2^{n-1}$. Then from the above and (3.47) we have that

$$\mathrm{H}_\infty\left(S_{l+1} \mid \Lambda, \overline{W}_1, \ldots, \overline{W}_{j-1}, \overline{C}, \overline{B}\right) \geq n - 1 - \lambda\,. \tag{3.48}$$

To cater for the fact that S_{l+1} is truncated to the leftmost τ bits, we apply Lemma 2.3.8 once more. The probability of guessing $\lceil S_{l+1} \rceil^\tau$ can only increase when given the rightmost $n - \tau$ bits of S_{l+1}. Thus

$$\mathrm{H}_\infty\left(\lceil S_{l+1} \rceil^\tau \mid \Lambda, \overline{W}_1, \ldots, \overline{W}_{j-1}, \overline{C}, \overline{B}\right) \geq \tau - \lambda - 1\,. \tag{3.49}$$

Combining inequalities (3.45), (3.46), and (3.49) yields

$$\Pr[W \mid \overline{C}, \overline{B}] \leq \frac{q_{\mathsf{Guess}}}{2^{\tau-\lambda-1}}\,. \tag{3.50}$$

The theorem then follows by combining (3.37), (3.38), (3.44), and (3.50). □

Security of SPRG

As explained above, SLENC can be decomposed into the cascade of SLFUNC and SPRG, matching the encryption component of the FGHF′ construction. A pseudocode description of the variable-output-length pseudorandom generator SPRG is given in Figure 3.10 while it is illustrated in Figure 3.12. Decomposing SLENC this way requires us to treat all of SPRG's initial state as the seed, which deviates from the more conventional ways of constructing sponge-based pseudorandom generators as, for instance, in [BDPV10]. Moreover, we prove security in a case which allows the adversary to make multiple queries to the PRG, each with differing output lengths. Security for the standard case of distinguishing a single output of the pseudorandom generator from a random bit string follows trivially.

The security of SPRG is stated formally in Theorem 3.2.3 below. Its proof follows from a standard hybrid argument.

Theorem 3.2.3. *Let* SPRG *be the pseudorandom generator described in Figure 3.10. Then, for any* PRG *adversary* $\mathcal{A}$*, it holds that*

$$\mathbf{Adv}^{\mathsf{PRG}}_{\mathrm{SPRG}}(\mathcal{A}) \leq \frac{l q_{\mathsf{G}} q_\rho}{2^c - q_\rho} + \frac{l q_\rho q_{\mathsf{G}} + l^2 q_{\mathsf{G}}^2}{2^n} + \frac{q_{\mathsf{G}}^2}{2^n}\,.$$

In the above, $\mathcal{A}$ *makes at most* q_ρ *queries to the random transformation* ρ *and* q_{G} *queries to the challenge oracle* G*. Moreover,* $l = \lceil \frac{L_{max}}{r} \rceil$*, where* L_{max} *is an upper bound on the output length that* $\mathcal{A}$ *queries to* G*.*

Proof. The proof follows through a standard hybrid argument, where we gradually replace the SPRG construction with a random function that takes as its input the seed S. In each game hop we remove one call to the random transformation and extend the output of the random function. This is illustrated in Figure 3.18.

Let L_{max} be an upper bound on the output bit-length specified by the adversary in its queries to G. Then each evaluation will require at most $l - 1$ calls to the random

Game G

$\mathsf{win} \leftarrow 0$
$\mathcal{S} \leftarrow \emptyset$
$K \leftarrow_{\$} \mathcal{K}$
$\mathcal{A}^{\mathsf{Guess},\mathsf{F},\mathsf{Lkg},\rho}()$
return win

oracle $\rho(Z)$

if $p[Z] = \perp$
  $p[Z] \leftarrow_{\$} \{0,1\}^n$
return $p[Z]$

oracle $\mathsf{Guess}(X, Y')$

parse X **as** $X_1 \parallel \ldots \parallel X_l \ \forall i \ |X_i| = \zeta$
$Y_0 \leftarrow K \parallel IV$
if $p[Y_0] = \perp$
  $p[Y_0] \leftarrow_{\$} \{0,1\}^n$
$S_1 \leftarrow p[Y_0]$
for i **in** $\{1, \ldots, l\}$
  $Y_i \leftarrow (\bar{S}_i \oplus X_i) \parallel \hat{S}_i$
  if $p[Y_i] = \perp$
    $p[Y_i] \leftarrow_{\$} \{0,1\}^n$
  $S_{i+1} \leftarrow p[Y_i]$
$Y \leftarrow \lceil S_{l+1} \rceil^{\tau}$
if $X \notin \mathcal{S} \wedge Y = Y'$
  $\mathsf{win} \leftarrow \mathrm{true}$
return $(Y = Y')$

oracle $\mathsf{F}(X)$

$\mathcal{S} \leftarrow_{\cup} X$
parse X **as** $X_1 \parallel \ldots \parallel X_l \ \forall i \ |X_i| = \zeta$
$Y_0 \leftarrow K \parallel IV$
if $p[Y_0] = \perp$
  $p[Y_0] \leftarrow_{\$} \{0,1\}^n$
$S_1 \leftarrow p[Y_0]$
for i **in** $\{1, \ldots, l\}$
  $Y_i \leftarrow (\bar{S}_i \oplus X_i) \parallel \hat{S}_i$
  if $p[Y_i] = \perp$
    $p[Y_i] \leftarrow_{\$} \{0,1\}^n$
  $S_{i+1} \leftarrow p[Y_i]$
return $\lceil S_{l+1} \rceil^{\tau}$

oracle $\mathsf{Lkg}(X, L_0, \ldots, L_{l+1})$

parse X **as** $X_1 \parallel \ldots \parallel X_l \ \forall i \ |X_i| = \zeta$
$Y_0 \leftarrow K \parallel IV$
$\Lambda \leftarrow L_0(Y_0)$
if $p[Y_0] = \perp$
  $p[Y_0] \leftarrow_{\$} \{0,1\}^n$
$S_1 \leftarrow p[Y_0]$
for i **in** $\{1, \ldots, l\}$
  $Y_i \leftarrow (\bar{S}_i \oplus X_i) \parallel \hat{S}_i$
  $\Lambda \leftarrow \Lambda \parallel L_i(Y_i)$
  if $p[Y_i] = \perp$
    $p[Y_i] \leftarrow_{\$} \{0,1\}^n$
  $S_{i+1} \leftarrow p[Y_i]$
$\Lambda \leftarrow \Lambda \parallel L_{l+1}(S_{l+1})$
return (Λ)

Figure 3.17: Game G used in the proof of Theorem 3.2.2.

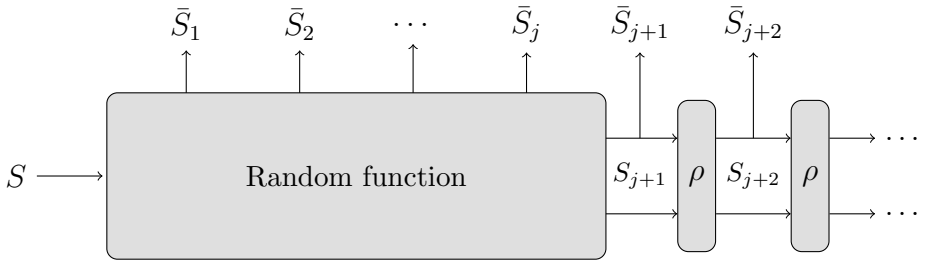

Figure 3.18: Graphical illustration of the hybrid game H_j used in the proof of Theorem 3.2.3. The large box represents a random function with input S and output $(\bar{S}_2, \ldots, \bar{S}_j, S_{j+1})$. The additional output $\bar{S}_1$ is simply copied from the input, i.e., $\bar{S}_1 = \bar{S}$.

transformation, where $l = \lceil \frac{L_{max}}{r} \rceil$. Accordingly, we specify the hybrid games $\mathsf{H}_0, \ldots, \mathsf{H}_{l-1}$, as shown in Figure 3.19. Following the hybrid argument, a further game hop to game G_0, displayed in Figure 3.20, is required to complete the proof.

In game H_j (cf. Figure 3.19), the oracle G samples the states $S_1, \ldots, S_{j+1}$ ideally and independent of the transformation ρ, while the remaining states $S_{j+2}, \ldots, S_l$ are sampled by querying ρ. The output of oracle G is the concatenation of the outer states truncated to L bits. The boxed version, that is, $\boxed{\mathsf{H}_j}$, ensures that the state S_{j+1} is also evaluated through ρ, and thus $\mathsf{H}_j = \boxed{\mathsf{H}_{j+1}}$. Note that H_0 is equivalent to the PRG game instantiated with SPRG with the challenge bit b fixed to 1. Thus,

$$\Pr[\mathcal{A}^{\mathsf{PRG}} \Rightarrow 1 \,|\, b = 1] = \Pr[\mathcal{A}^{\mathsf{H}_0} \Rightarrow 1] \,. \tag{3.51}$$

Moreover, we have that

$$\Pr[\mathcal{A}^{\mathsf{H}_0} \Rightarrow 1] - \Pr[\mathcal{A}^{\mathsf{H}_{l-1}} \Rightarrow 1] = \sum_{j=1}^{l-1} \Big(\Pr[\mathcal{A}^{\mathsf{H}_{j-1}} \Rightarrow 1] - \Pr[\mathcal{A}^{\mathsf{H}_j} \Rightarrow 1] \Big) \,,$$

and since $\mathsf{H}_{j-1} = \boxed{\mathsf{H}_j}$, we obtain

$$\sum_{j=1}^{l-1} \Big(\Pr[\mathcal{A}^{\mathsf{H}_{j-1}} \Rightarrow 1] - \Pr[\mathcal{A}^{\mathsf{H}_j} \Rightarrow 1] \Big) = \sum_{j=1}^{l-1} \Big(\Pr[\mathcal{A}^{\boxed{\mathsf{H}_j}} \Rightarrow 1] - \Pr[\mathcal{A}^{\mathsf{H}_j} \Rightarrow 1] \Big) \,.$$

Let Bad_j^1, Bad_j^2, and Bad_j^3 each denote the event that the corresponding flag in game $\boxed{\mathsf{H}_j}$ is set. Further note that games $\boxed{\mathsf{H}_j}$ and H_j are identical until one of these bad events occurs. Then, applying the fundamental lemma of game playing and the union bound yields

$$\begin{aligned} \sum_{j=1}^{l-1} \Big(\Pr[\mathcal{A}^{\boxed{\mathsf{H}_j}} \Rightarrow 1] - \Pr[\mathcal{A}^{\mathsf{H}_j} \Rightarrow 1] \Big) &\leq \sum_{j=1}^{l-1} \Pr[\mathsf{Bad}_j^1 \cup \mathsf{Bad}_j^2 \cup \mathsf{Bad}_j^3] \\ &\leq \sum_{j=1}^{l-1} \big(\Pr[\mathsf{Bad}_j^1] + \Pr[\mathsf{Bad}_j^2] + \Pr[\mathsf{Bad}_j^3] \big) \,. \end{aligned} \tag{3.52}$$

The flag Bad_j^1 is set if the adversary makes a query Z to ρ, such that $Z = S_j[S^*]$ for some $S^* \in \mathcal{D}$. Now, for any random variable $S_j[S^*]$, only the outer part $\bar{S}_j[S^*]$ is known to the adversary (through the oracle G) and the inner c bits remain hidden. Moreover, at any point in time, there are only $|\mathcal{D}|$ such variables and $|\mathcal{D}| \leq q_{\mathsf{G}}$. Hence, by the union bound

$$\Pr[\mathsf{Bad}_j^1] \leq \sum_{i=0}^{q_\rho - 1} \frac{|\mathcal{D}|}{2^c - i} \leq \sum_{i=0}^{q_\rho - 1} \frac{q_{\mathsf{G}}}{2^c - i} \leq \frac{q_{\mathsf{G}} q_\rho}{2^c - q_\rho} \,. \tag{3.53}$$

The flag Bad_j^2 is set whenever the oracle G samples a value S_j contained in $\mathsf{inset}(p)$. Since the oracle G samples the values S_j uniformly at random from $\{0,1\}^n$, it follows that

$$\Pr[\mathsf{Bad}_j^2] \leq \sum_{i=1}^{q_{\mathsf{G}}} \frac{|\mathsf{inset}(p)|}{2^n} \,.$$

At any point in time, $|\mathsf{inset}(p)| \leq q_\rho + q_{\mathsf{G}}(l - j - 1)$, since each query to ρ and G adds at most 1 and $(l - j - 1)$, new values to $\mathsf{inset}(p)$ respectively. This leads to

$$\sum_{i=1}^{q_{\mathsf{G}}} \frac{|\mathsf{inset}(p)|}{2^n} \leq \sum_{i=1}^{q_{\mathsf{G}}} \frac{q_\rho + q_{\mathsf{G}}(l - j - 1)}{2^n} \leq \frac{q_\rho q_{\mathsf{G}} + q_{\mathsf{G}}^2 (l - j - 1)}{2^n} \,. \tag{3.54}$$

The flag Bad_j^3 is set if, for some $i \in \{j + 1, \ldots, l - 1\}$ and some $S^* \in \mathcal{D}$, oracle G samples a value $S_i[S]$ equal to $S_i[S^*]$. The probability of this bad event depends solely on the number of queries that $\mathcal{A}$ makes to its oracle G. In each query at most, $(l - j - 1)$ states are sampled which can set this flag, hence

$$\Pr[\mathsf{Bad}_j^3] \leq \sum_{i=1}^{q_{\mathsf{G}}} \frac{(l - j - 1)|\mathcal{D}|}{2^n} \,.$$

Since at any point in time $|\mathcal{D}| \leq q_{\mathsf{G}}$, it holds that

$$\sum_{i=1}^{q_{\mathsf{G}}} \frac{(l - j - 1)|\mathcal{D}|}{2^n} \leq \sum_{i=1}^{q_{\mathsf{G}}} (l - j - 1) \frac{q_{\mathsf{G}}}{2^n} = \frac{(l - j - 1) q_{\mathsf{G}}^2}{2^n} \,. \tag{3.55}$$

Combining (3.52), (3.53), (3.54), and (3.55) yields

$$\begin{aligned}
\Pr[\mathcal{A}^{\mathsf{H}_0} \Rightarrow 1] - \Pr[\mathcal{A}^{\mathsf{H}_{l-1}} \Rightarrow 1] &\leq \sum_{j=1}^{l-1} \left(\Pr[\mathsf{Bad}_j^1] + \Pr[\mathsf{Bad}_j^2] + \Pr[\mathsf{Bad}_j^3] \right) \\
&\leq \sum_{j=1}^{l-1} \left(\frac{q_{\mathsf{G}} q_\rho}{2^c - q_\rho} + \frac{q_\rho q_{\mathsf{G}} + q_{\mathsf{G}}^2 (l - j - 1)}{2^n} + \frac{(l - j - 1) q_{\mathsf{G}}^2}{2^n} \right) \\
&\leq \frac{l q_{\mathsf{G}} q_\rho}{2^c - q_\rho} + \frac{l q_\rho q_{\mathsf{G}} + l^2 q_{\mathsf{G}}^2}{2^n} \,.
\end{aligned} \tag{3.56}$$

To conclude the proof, we require one more game hop. Consider the game $\boxed{\mathsf{G}_0}$ displayed in Figure 3.20. In this game, the oracle G first samples a value S, followed by sampling a

random output Z, independent of ρ. The boxed code ensures that the output Z is identical if a value for S is sampled twice. This game is equivalent to H_{l-1} which yields

$$\Pr[\mathcal{A}^{\mathsf{H}_{l-1}} \Rightarrow 1] = \Pr[\mathcal{A}^{\boxed{\mathsf{G}_0}} \Rightarrow 1]\,. \tag{3.57}$$

Now, game $\boxed{\mathsf{G}_0}$ and game G_0 (also displayed in Figure 3.20) are identical until Bad occurs. The flag is set to true if a value S is sampled that is already in the set $\mathcal{D}$. At any point in time $|\mathcal{D}| \leq q_{\mathsf{G}}$, and S is sampled by oracle G independent of $\mathcal{A}$, thus

$$\Pr[\mathcal{A}^{\boxed{\mathsf{G}_0}} \Rightarrow 1] - \Pr[\mathcal{A}^{\mathsf{G}_0} \Rightarrow 1] \leq \Pr[\mathsf{Bad}] \leq \sum_{i=1}^{q_{\mathsf{G}}} \frac{|\mathcal{D}|}{2^n} \leq \sum_{i=1}^{q_{\mathsf{G}}} \frac{q_{\mathsf{G}}}{2^n} \leq \frac{q_{\mathsf{G}}^2}{2^n}\,. \tag{3.58}$$

Moreover, G_0 and game PRG, with challenge bit fixed to 0, are functionally equivalent, and therefore

$$\Pr[\mathcal{A}^{\mathsf{G}_0} \Rightarrow 1] = \Pr[\mathcal{A}^{\mathsf{PRG}} \Rightarrow 1 \,|\, b = 0]\,. \tag{3.59}$$

By combining equations (3.51), (3.56), (3.57), (3.58), and (3.59) we obtain

$$\begin{aligned}
\mathbf{Adv}^{\mathsf{PRG}}_{\mathrm{SPRG}}(\mathcal{A}) &= \Pr[\mathcal{A}^{\mathsf{PRG}} \Rightarrow 1 \,|\, b = 1] - \Pr[\mathcal{A}^{\mathsf{PRG}} \Rightarrow 1 \,|\, b = 0] \\
&= \Pr[\mathcal{A}^{\mathsf{H}_0} \Rightarrow 1] - \Pr[\mathcal{A}^{\mathsf{G}_0} \Rightarrow 1] \\
&= \Pr[\mathcal{A}^{\mathsf{H}_0} \Rightarrow 1] - \Pr[\mathcal{A}^{\mathsf{H}_{l-1}} \Rightarrow 1] + \Pr[\mathcal{A}^{\mathsf{H}_{l-1}} \Rightarrow 1] - \Pr[\mathcal{A}^{\mathsf{G}_0} \Rightarrow 1] \\
&\leq \frac{l q_{\mathsf{G}} q_\rho}{2^c - q_\rho} + \frac{l q_\rho q_{\mathsf{G}} + l^2 q_{\mathsf{G}}^2}{2^n} + \frac{q_{\mathsf{G}}^2}{2^n}\,.
\end{aligned}$$

This concludes the proof. □

Security of SvHash

The final building block is the sponge-based vector hash function SvHash, which is graphically represented in Figure 3.13. It takes as input a triple of strings, namely a nonce, associated data, and a ciphertext to return a string digest. A salient feature of this construction is the XORing of $1 \,\|\, 0^{c-1}$ into the inner state in order to separate the (padded) associated data from the (padded) ciphertext. We prove the security of SvHash in a modular fashion, by first reducing its security to that of a plain hash function taking a single input and then prove the collision resistance of this latter construction in the random transformation model. The collision resistance of SvHash is stated formally in the following theorem.

Theorem 3.2.4. *Let* SvHash *be the vector hash function described in Figure 3.10. Then for any adversary $\mathcal{A}$ making q queries to ρ, it holds that*

$$\mathbf{Adv}^{\mathsf{CR}}_{\mathrm{SvHash}}(\mathcal{A}) \leq \frac{q(q-1)}{2^{w+1}} + \frac{q(q+2)}{2^{c-1}}\,.$$

Proof. We prove the theorem in two stages. First, we reduce the collision resistance of the vector hash function to the collision resistance of the standard sponge-based hash

Games $\mathsf{H}_j, \boxed{\mathsf{H}_j}$

$b \leftarrow\!\!\$\ \{0,1\}$
$b' \leftarrow \mathcal{A}^{\mathsf{G},\rho}()$

oracle $\rho(Z)$

if $p[Z] = \bot$
  $p[Z] \leftarrow\!\!\$\ \{0,1\}^n$
  // keep random function and ρ consistent
  if $\exists S^* \in \mathcal{D}$ **s.t.** $Z = S_j[S^*]$
    $\mathsf{Bad}_j^1 \leftarrow$ true
    $\boxed{p[Z] \leftarrow S_{j+1}[S^*]}$
return $p[Z]$

oracle $\mathsf{G}(L)$

$S \leftarrow\!\!\$\ \{0,1\}^n$
$\mathcal{D} \leftarrow\!\!\cup\ S$
$S_1[S] \leftarrow S$
// evaluate random function
for i **in** $\{2, \ldots, j+1\}$
  if $S_i[S] = \bot$
    $S_i[S] \leftarrow\!\!\$\ \{0,1\}^n$
// keep random function and ρ consistent
if $S_j[S] \in \mathsf{inset}(p)$
  $\mathsf{Bad}_j^2 \leftarrow$ true
  $\boxed{S_{j+1}[S] \leftarrow p[S_j[S]]}$
// evaluate remaining part of SPRG
for i **in** $\{j+1, \ldots, l-1\}$
  if $p[S_i[S]] = \bot$
    $p[S_i[S]] \leftarrow\!\!\$\ \{0,1\}^n$
    // keep random function and ρ consistent
    if $\exists S^* \in \mathcal{D}$ **s.t.** $S_i[S] = S_j[S^*]$
      $\mathsf{Bad}_j^3 \leftarrow$ true
      $\boxed{p[S_i[S]] \leftarrow S_{j+1}[S^*]}$
  $S_{i+1}[S] \leftarrow p[S_i[S]]$
$Z \leftarrow \bar{S}_1[S] \parallel \bar{S}_2[S] \parallel \ldots \parallel \bar{S}_l[S]$
return $\lceil Z \rceil^L$

Figure 3.19: Hybrid games H_j and $\boxed{\mathsf{H}_j}$ used in the proof of Theorem 3.2.3. The boxed code is included in $\boxed{\mathsf{H}_j}$ whereas in H_j it is not.

Games $\mathsf{G}_0, \boxed{\mathsf{G}_0}$

$b \leftarrow\!\!\$\ \{0,1\}$
$b' \leftarrow \mathcal{A}^{\mathsf{G},\rho}()$

oracle $\rho(Z)$

if $p[Z] = \bot$
  $p[Z] \leftarrow\!\!\$\ \{0,1\}^n$
return $p[Z]$

oracle $\mathsf{G}(L)$

$S \leftarrow\!\!\$\ \{0,1\}^n$
$Z \leftarrow\!\!\$\ \{0,1\}^{lr}$
if $S \in \mathcal{D}$
  $\mathsf{Bad} \leftarrow$ true
  $\boxed{Z \leftarrow \bar{S}_1[S] \parallel \ldots \parallel \bar{S}_l[S]}$
$\mathcal{D} \leftarrow\!\!\cup\ S$
$(\bar{S}_1[S], \ldots, \bar{S}_l[S]) \leftarrow Z$
return $\lceil Z \rceil^L$

Figure 3.20: Games G_0 and $\boxed{\mathsf{G}_0}$ used in the proof of Theorem 3.2.3. The boxed code is included in $\boxed{\mathsf{G}_0}$ but not in G_0.

function. Towards this end we introduce a padding scheme which we call "input separation padding" (isPad). This will allow us to view the evaluation of the vector hash over a triple of inputs as the evaluation of the standard sponge-based hash over an encoding of this triple of inputs into a single string—see Figure 3.21. We show that isPad is injective, and consequently that any collision in the vector hash yields a collision in the standard sponge-based hash function. In the second part, we prove that the standard sponge-based hash function, instantiated with a random transformation, is collision-resistant. This part is based on the proof described by Boneh and Shoup in [BS20]. However, since their proof is in the random permutation model, it requires some changes to adapt it to the random transformation model to deal with collisions of the underlying transformation, which are non-existent in their case.

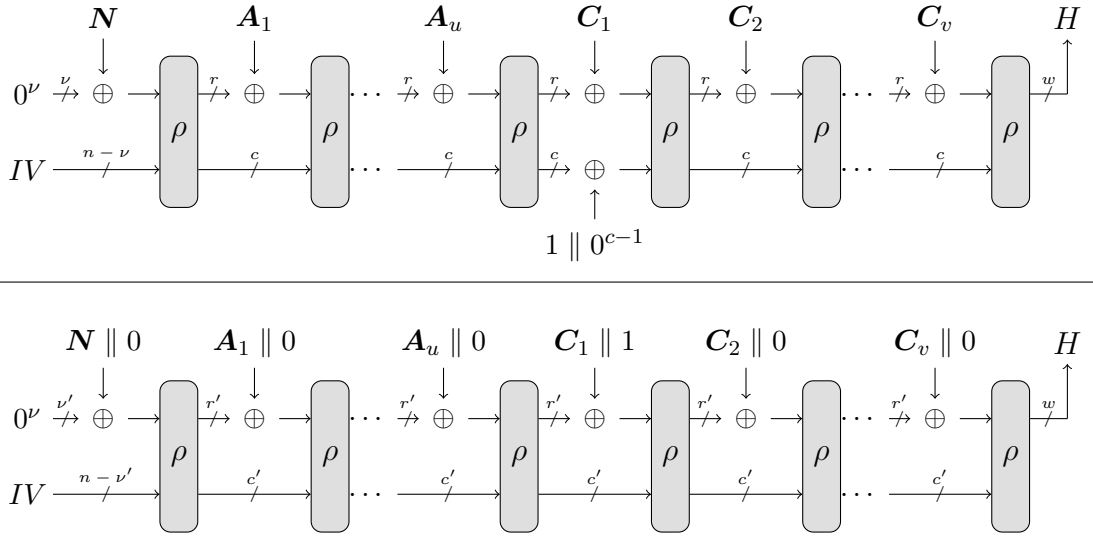

Figure 3.21: **Above:** the sponge-based vector hash function SvHash that is used in Slae. **Below:** the sponge-based vector hash function SvHash viewed as the standard sponge-based hash function SHash with its inputs encoded via isPad.

The padding isPad is described in Figure 3.22, which in turn makes use of $\mathsf{lpad}^\star$. The latter takes the triple (N, A, C) and an integer r (the rate) as input and outputs $(\boldsymbol{N}, \boldsymbol{A}, \boldsymbol{C})$ such that the size of each is a multiple of r. The first input (N) is simply padded with 0's to a length r, whereas A and C are padded with $\mathsf{lpad}(A, r)$ and $\mathsf{lpad}(C, r)$. Here lpad simply applies the 10^* padding described in Figure 3.10 and is already used in SvHash, see Figure 3.13. Then $(\boldsymbol{N}, \boldsymbol{A}, \boldsymbol{C})$ is mapped to a string $Z = Z_1 \parallel Z_2 \parallel Z_3$ whose length is a multiple of $r + 1$, where Z_1 is $\boldsymbol{N}$ appended with a 0, Z_2 is $\boldsymbol{A}$ with a 0 inserted after each r bits, and Z_3 is $\boldsymbol{C}$ where a 1 is inserted after the first r bits and a 0 after all other r bits. Note that nonces are assumed to be of fixed size.

First note that lpad is itself an injective encoding. Assume now, towards a contradiction, that the isPad encodings of two distinct triples are equal, i.e., $(N, A, C) \neq (N', A', C')$ and $Z = \overline{Z}$. It then immediately follows that $N = N'$, since the nonces are of fixed size, and that $|Z| = |\overline{Z}|$. There are then only two possible cases: (1) $|Z_2| \neq |\overline{Z}_2|$ or (2) $|Z_2| = |\overline{Z}_2|$.

For the first case, assume, without loss of generality, that $|Z_2| < |\overline{Z}_2|$, i.e., $u < \overline{u}$, where u and $\overline{u}$ are the number of r-bit blocks of $\boldsymbol{A}$ and $\overline{\boldsymbol{A}}$, respectively. Then, when writing the strings Z and $\overline{Z}$ directly below one another, the block $\boldsymbol{C}_1$ is above $\overline{\boldsymbol{A}}_{u+1}$. However, isPad requires that $\boldsymbol{C}_1$ is followed by a 1 whereas $\overline{\boldsymbol{A}}_{u+1}$ is followed by a 0 which contradicts the assumption that $Z = \overline{Z}$. As for the second case, it follows that $Z_2 = \overline{Z}_2$ and $Z_3 = \overline{Z}_3$. In turn this means that $\boldsymbol{A} = \overline{\boldsymbol{A}}$ and $\boldsymbol{C} = \overline{\boldsymbol{C}}$, and by the injectivity of lpad it also follows that $A = A'$ and $C = C'$. Since we already established that $N = N'$, we have that $(N, A, C) = (N', A', C')$, which contradicts our assumption that the inputs are distinct.

$\mathsf{isPad}((N, A, C), r)$

$(\boldsymbol{N}, \boldsymbol{A}, \boldsymbol{C}) \leftarrow \mathsf{lpad}^\star((N, A, C), r)$
$Z_1 \leftarrow \boldsymbol{N} \parallel 0$
$Z_2 \leftarrow \boldsymbol{A}_1 \parallel 0 \parallel \boldsymbol{A}_2 \parallel 0 \parallel \cdots \parallel \boldsymbol{A}_u \parallel 0$ **s.t.** $|\boldsymbol{A}_i| = r, \forall i$
$Z_3 \leftarrow \boldsymbol{C}_1 \parallel 1 \parallel \boldsymbol{C}_2 \parallel 0 \parallel \cdots \parallel \boldsymbol{C}_v \parallel 0$ **s.t.** $|\boldsymbol{C}_i| = r, \forall i$
return $Z \leftarrow Z_1 \parallel Z_2 \parallel Z_3$

$\mathsf{lpad}^\star((N, A, C), r)$

$\boldsymbol{N} \leftarrow N \parallel 0^{r-|N|}$
$\boldsymbol{A} \leftarrow A \parallel \mathsf{lpad}(A, r)$
$\boldsymbol{C} \leftarrow C \parallel \mathsf{lpad}(C, r)$
return $(\boldsymbol{N}, \boldsymbol{A}, \boldsymbol{C})$

Figure 3.22: The padding schemes isPad and $\mathsf{lpad}^\star$.

Now we can use isPad to reduce the collision resistance of SvHash over triples to the collision resistance of the standard sponge-based hash SHash over strings. Both hash functions are depicted in Figure 3.21. Furthermore, note that SvHash can be expressed as $\text{SvHash}(N, A, C) = \text{SHash}(\mathsf{isPad}(N, A, C))$. This mapping is such that SHash requires an extra call to the random transformation and has capacity $c' = c - 1$, where c is the capacity of SHash. Now, since isPad is injective, any collision that is found on SvHash must correspond to a collision on SHash. Moreover, since isPad is efficiently computable, any collision on SvHash can easily be translated into a collision on SHash. This leads to a straightforward reduction, which for any adversary $\mathcal{A}$ against SvHash yields an adversary with similar resources $\overline{\mathcal{A}}$ against SHash such that

$$\mathbf{Adv}^{\mathsf{CR}}_{\text{SvHash}}(\mathcal{A}) \leq \mathbf{Adv}^{\mathsf{CR}}_{\text{SHash}}(\overline{\mathcal{A}})\,. \tag{3.60}$$

This concludes the first part of the proof and we now go on to prove the collision resistance of the standard sponge-based hash function SHash with capacity c'.

We assume, without loss of generality, that the adversary makes queries to the random transformation which correspond to its final output. That is, it queries the random transformation on all intermediate states that occur while computing the output. Furthermore, we assume that no redundant queries are made by the adversary, i.e., whenever the adversary makes a query to ρ on Y yielding $S = \rho(Y)$, the adversary will never make another query to ρ on Y. This assumption is justified by the fact that an adversary gets no additional information from redundant queries and any adversary that makes redundant queries can easily be turned into an adversary which makes no redundant queries.

Similar to the proof in [BS20], we use a directed graph G to visualize the attack. At the start, the graph contains all bit strings of length n as nodes and no edges. Let Y and S be two nodes. An edge from Y to S is added to the graph, if the adversary makes a query to ρ on Y resulting in S. Since there are no redundant queries, an edge is added to the

graph exactly once. In contrast to the proof in [BS20], we have to deal with the fact that the transformation ρ is not injective and that a node may have more than one incoming edge. For $k \geq 1$, a path of length k can be described by a sequence of $2k$ nodes

$$Y_1, S_2, Y_2, S_3, Y_3, \ldots, S_k, Y_k, S_{k+1}\,.$$

Moreover, it holds that $\hat{Y}_1 = IV$, $\hat{S}_i = \hat{Y}_i$ for $i = 2, \ldots, k$, and the graph G contains edges $Y_i \to S_{i+1}$ for $i = 1, \ldots, k$. The message of this path is defined as a tuple $(M_1, \ldots, M_k)$ of bit strings of length r, where $M_1 = \bar{Y}_1$ and $M_i = \bar{S}_i \oplus \bar{Y}_i$ for $i = 2, \ldots, k$. The result of this path is $M_{k+1} = \bar{S}_{k+1}$. Such a path corresponds to the computation of $\text{SHASH}(M_1 \parallel \ldots \parallel M_k)$ resulting in output M_{k+1}. We write such paths as

$$M_1|Y_1 \to S_2|M_2|Y_2 \to \cdots \to S_k|M_k|Y_k \to S_{k+1}|M_{k+1}\,.$$

We can then define a collision in the hash function in terms of the graph G. A collision corresponds to finding a pair of "colliding paths", which are paths on different messages $(M_1, \ldots, M_k) \neq (M'_1, \ldots, M'_l)$ resulting in messages M_{k+1} and M'_{l+1} which agree in their first w bits, i.e., $\lceil M_{k+1} \rceil^w = \lceil M'_{l+1} \rceil^w$.

As in [BS20], we use the notion of "problematic paths". Consider the following two paths on messages $(M_1, \ldots, M_k) \neq (M'_1, \ldots, M'_l)$ of length k and l, respectively.

$$\begin{aligned} M_1|Y_1 \to S_2|M_2|Y_2 \to \cdots \to S_{k-1}|M_{k-1}|Y_{k-1} \to S_k|M_k|Y_k \to S_{k+1}|M_{k+1} \\ M'_1|Y'_1 \to S'_2|M'_2|Y'_2 \to \cdots \to S'_{l-1}|M'_{l-1}|Y'_{l-1} \to S'_l|M'_l|Y'_l \to S'_{l+1}|M'_{l+1} \end{aligned}$$

Intuitively, we call a pair of paths *problematic* if their states are equal before the last random transformation is applied. More formally, using the above representation of a path, if $Y_k = Y'_l$ then the paths are called problematic.

Let us denote by CP the event that the adversary finds a pair of colliding paths and by PP the event that the adversary finds a pair of problematic paths. Then the probability of finding a pair of colliding paths, i.e., a collision in the hash function, can be bounded as follows

$$\begin{aligned} \Pr[CP] &= \Pr[CP \cap \overline{PP}] + \Pr[CP \cap PP] \\ &\leq \Pr[CP \cap \overline{PP}] + \Pr[PP]\,. \end{aligned} \tag{3.61}$$

We start by proving an upper bound for $\Pr[CP \cap \overline{PP}]$. This argument closely follows the argument given in [BS20], however, we obtain a different bound due to the fact that our proof is in the random transformation model. A pair of paths are both colliding and not problematic means that the final edges correspond to queries to ρ on different inputs resulting in outputs where the first w bits are equal. That is, the adversary makes queries to ρ on $Y \neq Y'$ which result in S and S', respectively, such that $\lceil \bar{S} \rceil^w = H = \lceil \bar{S}' \rceil^w$.

For $i \leq j$, let X_{ij} denote the event that the i^{th} query of the adversary is some value Y resulting in $S = \rho(Y)$, while the j^{th} query is on some value $Y' \neq Y$, yielding output $S' = \rho(Y')$ with $\lceil \bar{S} \rceil^w = \lceil \bar{S}' \rceil^w$. Fix i, j, the random coins of the adversary, and the outputs of all queries made before the j^{th} query. Then Y, S, and Y' are fixed while S' is

distributed uniformly at random over a set of size 2^n. Since there are 2^{n-w} nodes W for which it holds that $\lceil W \rceil^w = \lceil \bar{S} \rceil^w$, S' must be equal to one of these, thus

$$\Pr[X_{ij}] \leq \frac{2^{n-w}}{2^n} \leq \frac{1}{2^w}\,,$$

which leads to

$$\Pr[CP \cap \overline{PP}] \leq \sum_{j=1}^{q} \sum_{i=1}^{j-1} \Pr[X_{ij}] \leq \sum_{j=1}^{q} \sum_{i=1}^{j-1} \frac{1}{2^w} \leq \frac{q(q-1)}{2^{w+1}}\,.$$

Next, we prove an upper bound for $\Pr[PP]$. The probability of finding a pair of problematic paths is closely related to two basic attacks against random sponges introduced by Bertoni et al. [BDPVA07], which allows us to upper bound the probability. The first one, called "path to an inner state", asks, for a given bit string $x \in \{0,1\}^{c'}$, to find a path to a node S with inner part equal to x, i.e., $\hat{S} = x$. To match this to our case, we define $E1$ to be the event that an adversary finds a path to the inner state IV. The second one, called "inner collision", asks to find two different paths to nodes S and S' with equal inner states, i.e., $\hat{S} = \hat{S}'$, which we define to be $E2$. In the following, we show that

$$\Pr[PP] \leq \Pr[E1] + \Pr[E2]\,, \tag{3.62}$$

i.e., every adversary that finds a pair of problematic paths either finds a path to the inner state IV ($E1$) or an inner collision ($E2$). Consider an adversary that finds a pair of problematic paths which are of length k and l, respectively. Write these paths as

$$\begin{aligned} M_1|Y_1 \to S_2|M_2|Y_2 \to \cdots \to S_{k-1}|M_{k-1}|Y_{k-1} \to S_k|M_k|Y_k \to S_{k+1}|M_{k+1} \\ M'_1|Y'_1 \to S'_2|M'_2|Y'_2 \to \cdots \to S'_{l-1}|M'_{l-1}|Y'_{l-1} \to S'_l|M'_l|Y'_l \to S'_{l+1}|M'_{l+1}\,. \end{aligned}$$

Due to the fact that the paths are problematic, it holds that both $Y_k = Y'_l$ as well as $(M_1, \ldots, M_k) \neq (M'_1, \ldots, M'_l)$. We further assume that the pair is the shortest among all pairs of problematic paths. This means that $k + l$ is minimal, and without loss of generality, that $k \leq l$. There are three cases to consider, depending on the length of the paths:

1. ($k = 1$ and $l = 1$). In this case, the paths are simply $M_1|Y_1 \to S_2|M_2$ and $M'_1|Y'_1 \to S'_2|M'_2$, the messages of these paths are M_1 and M'_1, respectively. However, it is impossible that both $Y_1 = Y'_1$ and $M_1 \neq M'_1$. By construction it holds that

   $$\bar{Y}_1 = M_1 \neq M'_1 = \bar{Y}'_1$$

   which leads to a contradiction.

2. ($k = 1$ and $l \geq 2$). In this case, we have that $Y_1 = Y'_l$. The adversary has made a query to ρ on Y'_{l-1} which resulted in $S'_l = \rho(Y'_{l-1})$. Therefore, it holds that $\hat{S}'_l = \hat{Y}'_l = \hat{Y}_1 = IV$. The first equality follows from the construction, where the inner state between two evaluations of ρ does not change. The second one trivially follows from $Y_1 = Y'_l$, while the third follows again from the construction. By taking the path up to node S'_l, the adversary has found a path to the inner state IV, i.e., event $E1$ occurs.

3. ($k \geq 2$ and $l \geq 2$). Here it holds that $Y_k = Y'_l$. Consider now the last but one edges which are

$$Y_{k-1} \to S_k | M_k | Y_k$$
$$Y'_{l-1} \to S'_l | M'_l | Y'_l$$

There are two cases depending on the absorbed messages M_k and M'_l:

a) The absorbed messages are equal, i.e., $M_k = M'_l$. Hence, it holds that $S_k = S'_l$. Due to the fact that ρ is a random transformation we have to further distinguish between the following sub-cases:
   i. ($Y_{k-1} = Y'_{l-1}$). This contradicts the minimality of $k + l$, as we can find a shorter pair of problematic paths by throwing away the last edge in each path, while the truncated messages $(M_1, \ldots, M_{k-1})$ and $(M'_1, \ldots, M'_{l-1})$ still differ.
   ii. ($Y_{k-1} \neq Y'_{l-1}$). In this case, both Y_{k-1} and Y'_{l-1} map to $S_k = S'_l$, therefore the adversary has found an inner collision on the nodes S_k and S'_l, i.e., event $E2$ occurs.

b) The absorbed messages are different, i.e., $M_k \neq M'_l$. This yields $\bar{S}_k \neq \bar{S}'_l$ which further implies $S_k \neq S'_l$ and $Y_{k-1} \neq Y'_{l-1}$. On the other hand, the absorbed messages do not affect the inner state of S_k and S'_l, which ensures $\hat{S}_k = \hat{S}'_l$. Combining these, it holds that the adversary has found an inner collision on the nodes S_k and S'_l by taking the paths up to S_k and S'_l, i.e., event $E2$ occurs. We emphasise that $Y_{k-1} \neq Y'_{l-1}$ implies that the paths are indeed distinct.

This proves inequality (3.62). Thus, we can upper bound $\Pr[PP]$ using the following bounds on $\Pr[E1]$ and $\Pr[E2]$ by Bertoni et al. [BDPVA07], where q denotes the number of queries the adversary makes to ρ:

$$\Pr[E1] \leq \frac{q}{2^{c'}} \qquad \text{and} \qquad \Pr[E2] \leq \frac{q(q+1)}{2^{c'+1}}\,.$$

Substituting everything into (3.61) yields

$$\Pr[CP] \leq \frac{q(q-1)}{2^{w+1}} + \frac{q}{2^{c'}} + \frac{q(q+1)}{2^{c'+1}} \leq \frac{q(q-1)}{2^{w+1}} + \frac{q(q+2)}{2^{c'}}\,. \tag{3.63}$$

By combining (3.60) and (3.63) we get

$$\mathbf{Adv}^{\mathsf{CR}}_{\text{SvHash}}(\mathcal{A}) \leq \mathbf{Adv}^{\mathsf{CR}}_{\text{SHash}}(\overline{\mathcal{A}}) \leq \Pr[CP] \leq \frac{q(q-1)}{2^{w+1}} + \frac{q(q+2)}{2^{c'}}\,. \tag{3.64}$$

Finally, by replacing $c' = c - 1$ to cater for the domain separation of SvHash, we obtain the desired result

$$\mathbf{Adv}^{\mathsf{CR}}_{\text{SvHash}}(\mathcal{A}) \leq \frac{q(q-1)}{2^{w+1}} + \frac{q(q+2)}{2^{c-1}}\,,$$

which proves the theorem. □

3.3 LAE from LPRF

The generic FGHF′ construction reduces the task of constructing leakage-resilient AEAD schemes to constructing leakage-resilient function families that are both pseudorandom and unpredictable. While the former notion is well established in the literature, the latter is new and, except SLFUNC, no constructions are known. Hence, we turn our focus towards constructing leakage-resilient AEAD schemes from function families that are leakage-resilient pseudorandom.

In Section 3.3.1 we show that the two security notions LPRF and LUF are incomparable by giving separation examples in both directions. We then scrutinise the N2 composition theorem by Barwell et al. [BMOS17] for a special type of symmetric encryption schemes and MACs in Section 3.3.2. In Section 3.3.3, we show that the encryption scheme and MAC underlying the FGHF′ construction satisfy these requirements. Finally, we give the improved composition theorem for the FGHF′ construction in Section 3.3.4.

3.3.1 Unpredictability and Pseudorandomness under Leakage

As mentioned above, LUF and LPRF security are incomparable, which we show here by providing two separating functions: the first is unpredictable but not pseudorandom under leakage, while the second is pseudorandom but not unpredictable under leakage.

Under Leakage: Unpredictability $\not\Rightarrow$ Pseudorandomness

We start with the simple case, that is, a function which is unpredictable but not pseudorandom under leakage.

Construction 3.3.1. *Let $\mathcal{F}_*\colon \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^t$ be a function family. Define the function family*

$$\begin{aligned}\mathcal{F}\colon \{0,1\}^k \times \{0,1\}^n &\to \{0,1\}^{t+1}\\ (K,X) &\mapsto 0 \parallel \mathcal{F}_*(K,X)\,.\end{aligned}$$

Lemma 3.3.2. *Let $\mathcal{F}_*$ be a function family with associated leakage set $\mathcal{L}_F$ and $\mathcal{F}$ be the function family constructed from $\mathcal{F}_*$ according to Construction 3.3.1 with the same leakage set. For any adversary $\mathcal{A}_{\mathsf{luf}}$ against $\mathcal{F}$, there exists an adversary $\overline{\mathcal{A}}_{\mathsf{luf}}$ against $\mathcal{F}_*$ such that*

$$\mathbf{Adv}^{\mathsf{LUF}}_{\mathcal{F}}(\mathcal{A}_{\mathsf{luf}}, \mathcal{L}_F) \leq \mathbf{Adv}^{\mathsf{LUF}}_{\mathcal{F}_*}(\overline{\mathcal{A}}_{\mathsf{luf}}, \mathcal{L}_F)\,.$$

Furthermore, there exists an adversary $\mathcal{A}_{\mathsf{lprf}}$ against $\mathcal{F}_*$, making q queries to F, such that

$$\mathbf{Adv}^{\mathsf{LPRF}}_{\mathcal{F}}(\mathcal{A}_{\mathsf{lprf}}, \mathcal{L}_F) = 1 - 2^{-q}\,.$$

Proof. For the first part, we construct the following adversary $\overline{\mathcal{A}}_{\mathsf{luf}}$. It runs $\mathcal{A}_{\mathsf{luf}}$ forwarding any query that $\mathcal{A}_{\mathsf{luf}}$ makes to either F or Lkg to its own oracles and sends the response back to $\mathcal{A}_{\mathsf{luf}}$, for a query to F, it additionally prepends a 0 to its response before sending it back to $\mathcal{A}_{\mathsf{luf}}$. When $\mathcal{A}_{\mathsf{luf}}$ makes a query (X, Y') to Guess, $\overline{\mathcal{A}}_{\mathsf{luf}}$ queries $(X, \lfloor Y' \rfloor_t)$ to its own oracle Guess and forwards the response to $\mathcal{A}_{\mathsf{luf}}$.

It holds that $\overline{\mathcal{A}}_{\mathsf{luf}}$ perfectly simulates game LUF for $\mathcal{A}_{\mathsf{luf}}$ and both adversaries query the same inputs to F, hence the set $\mathcal{S}$ is the same for both. Let (X, Y') be the query by $\mathcal{A}_{\mathsf{luf}}$ that sets its winning flag win to true. Then, it holds that $Y' = \mathcal{F}(K, X) = 0 \parallel \mathcal{F}_*(K, X)$ and $X \notin \mathcal{S}$. By construction, $\overline{\mathcal{A}}_{\mathsf{luf}}$ will query $(X, \lfloor Y' \rfloor_t)$ to Guess which will set the flag win to true since $\lfloor Y' \rfloor_t = \lfloor \mathcal{F}(K, X) \rfloor_t = \lfloor 0 \parallel \mathcal{F}_*(K, X) \rfloor_t = \mathcal{F}_*(K, X)$. Thus, we obtain

$$\mathbf{Adv}^{\mathsf{LUF}}_{\mathcal{F}}(\mathcal{A}_{\mathsf{luf}}, \mathcal{L}_F) \leq \mathbf{Adv}^{\mathsf{LUF}}_{\mathcal{F}_*}(\overline{\mathcal{A}}_{\mathsf{luf}}, \mathcal{L}_F)\,.$$

For the second part, we construct adversary $\mathcal{A}_{\mathsf{lprf}}$ as follows. It starts by choosing $X_1, \ldots, X_q \leftarrow\!\!\$\ \{0,1\}^n$ and queries them to F to obtain $Y_1, \ldots, Y_q$. If there exists $i \in [q]$ such that $\lceil Y_i \rceil^1 = 1$, $\mathcal{A}_{\mathsf{lprf}}$ outputs 0, otherwise, i.e., $\lceil Y_i \rceil^1 = 0$ for all $i \in [q]$, it outputs 1. If $b = 1$, it holds that $Y_i = \mathcal{F}(K, X_i) = 0 \parallel \mathcal{F}_*(K, X_i)$. It follows that $\lceil Y_i \rceil^1 = 0$ for all $i \in [q]$ and by construction of $\mathcal{A}_{\mathsf{lprf}}$ we obtain

$$\Pr[\mathcal{A}^{\mathsf{LPRF}}_{\mathsf{lprf}} \Rightarrow 1 \mid b = 1] = 1\,.$$

If $b = 0$, it holds that Y_i is randomly sampled from $\{0,1\}^{t+1}$ for all $i \in [q]$. Hence, the probability that $\lceil Y_i \rceil^1 = 0$ for all $i \in [q]$ holds with probability 2^{-q} and by construction of $\mathcal{A}_{\mathsf{lprf}}$ we obtain

$$\Pr[\mathcal{A}^{\mathsf{LPRF}}_{\mathsf{lprf}} \Rightarrow 1 \mid b = 0] = 2^{-q}\,.$$

Combining the above, we conclude with

$$\mathbf{Adv}^{\mathsf{LPRF}}_{\mathcal{F}}(\mathcal{A}_{\mathsf{lprf}}, \mathcal{L}_F) = \left|\Pr[\mathcal{A}^{\mathsf{LPRF}}_{\mathsf{lprf}} \Rightarrow 1 \mid b = 1] - \Pr[\mathcal{A}^{\mathsf{LPRF}}_{\mathsf{lprf}} \Rightarrow 1 \mid b = 0]\right| \leq 1 - 2^{-q}\,,$$

which finishes the proof. □

Under Leakage: Pseudorandomness ⇏ Unpredictability

Now we address the more complex part of the separation example and show that there are functions which are pseudorandom but not unpredictable under leakage. It is based on the following construction, which uses two function $\mathcal{F}_O$ and $\mathcal{F}_I$, where the subscripts O and I indicate the outer and inner function, respectively.

Construction 3.3.3. *Consider function families* $\mathcal{F}_O\colon \{0,1\}^s \times \{0,1\}^n \to \{0,1\}^t$ *and* $\mathcal{F}_I\colon \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^s$. *Define the function family*

$$\begin{aligned}\mathcal{F}\colon \{0,1\}^k \times \{0,1\}^n &\to \{0,1\}^t \\ (K, X) &\mapsto \mathcal{F}_O(\mathcal{F}_I(K, X), X)\,.\end{aligned}$$

The idea of Construction 3.3.3 is as follows. It uses some master key K and, for each input X, it derives a session key K_s using the inner function $\mathcal{F}_I$, i.e., $K_s = \mathcal{F}_I(K, X)$. The output Y is generated by the outer function $\mathcal{F}_O$ using the session key K_s and input X. The lemma below shows that the construction is pseudorandom but not unpredictable under leakage.

Lemma 3.3.4. *Let $\mathcal{F}_O$ and $\mathcal{F}_I$ be two function families with associated leakage sets $\mathcal{L}_O$ and $\mathcal{L}_I$, respectively. Let $\mathcal{F}$ be the function family constructed from $\mathcal{F}_O$ and $\mathcal{F}_I$ according to Construction 3.3.3 with associated leakage set $\mathcal{L}_F = \mathcal{L}_O \times \mathcal{L}_I$. Let further $\mathcal{L}_O = \{L_O\}$, where $L_O(K, X) = K$. For any adversary $\mathcal{A}_{\mathsf{lprf}}$ against $\mathcal{F}$, making q queries to F, there exist adversaries $\overline{\mathcal{A}}_{\mathsf{lprf}}$ and $\overline{\mathcal{A}}_{\mathsf{prf}}$ against $\mathcal{F}_I$ and $\mathcal{F}_O$, respectively, such that*

$$\mathbf{Adv}^{\mathsf{LPRF}}_{\mathcal{F}}(\mathcal{A}_{\mathsf{lprf}}, \mathcal{L}_F) \leq \mathbf{Adv}^{\mathsf{LPRF}}_{\mathcal{F}_I}(\overline{\mathcal{A}}_{\mathsf{lprf}}, \mathcal{L}_I) + q\,\mathbf{Adv}^{\mathsf{PRF}}_{\mathcal{F}_O}(\overline{\mathcal{A}}_{\mathsf{prf}})\,.$$

Furthermore, there exists an adversary $\mathcal{A}_{\mathsf{luf}}$ such that:

$$\mathbf{Adv}^{\mathsf{LUF}}_{\mathcal{F}}(\mathcal{A}_{\mathsf{luf}}, \mathcal{L}_F) = 1\,.$$

Proof. For the first part, we make use of the games G_0, G_1, and G_2 displayed in Figure 3.23. The games are constructed such that G_0 is equal to LPRF with secret bit $b = 1$ and G_2 is equal to LPRF with secret bit $b = 0$. Recall that the leakage set $\mathcal{L}_F$ of $\mathcal{F}$ is the Cartesian product of the leakage sets $\mathcal{L}_O$ of $\mathcal{F}_O$ and $\mathcal{L}_I$ of $\mathcal{F}_I$. Using a simple reformulation to the adversarial advantage yields

$$\begin{aligned}
\mathbf{Adv}^{\mathsf{LPRF}}_{\mathcal{F}}(\mathcal{A}_{\mathsf{lprf}}, \mathcal{L}_F) &= \left|\Pr[\mathcal{A}^{\mathsf{LPRF}}_{\mathsf{lprf}} \Rightarrow 1 \mid b = 1] - \Pr[\mathcal{A}^{\mathsf{LPRF}}_{\mathsf{lprf}} \Rightarrow 1 \mid b = 0]\right| \\
&= \left|\Pr[\mathcal{A}^{\mathsf{G}_0}_{\mathsf{lprf}} \Rightarrow 1] - \Pr[\mathcal{A}^{\mathsf{G}_2}_{\mathsf{lprf}} \Rightarrow 1]\right| \\
&\leq \left|\Pr[\mathcal{A}^{\mathsf{G}_0}_{\mathsf{lprf}} \Rightarrow 1] - \Pr[\mathcal{A}^{\mathsf{G}_1}_{\mathsf{lprf}} \Rightarrow 1]\right| \\
&\quad + \left|\Pr[\mathcal{A}^{\mathsf{G}_1}_{\mathsf{lprf}} \Rightarrow 1] - \Pr[\mathcal{A}^{\mathsf{G}_2}_{\mathsf{lprf}} \Rightarrow 1]\right|\,. \qquad (3.65)
\end{aligned}$$

For the game hop between G_0 and G_1, we construct an LPRF adversary $\overline{\mathcal{A}}_{\mathsf{lprf}}$ against $\mathcal{F}_I$ as follows. When $\mathcal{A}_{\mathsf{lprf}}$ queries X to F, $\overline{\mathcal{A}}_{\mathsf{lprf}}$ queries X to its own challenge oracle to obtain the session key K_s, computes $Y \leftarrow \mathcal{F}_O(K_s, X)$, and sends Y back to $\mathcal{A}_{\mathsf{lprf}}$. For leakage queries $(X, (L_O, L_I))$ by $\mathcal{A}_{\mathsf{lprf}}$, $\overline{\mathcal{A}}_{\mathsf{lprf}}$ queries its own leakage oracle on (X, L_I) to obtain (K_s, Λ_I), computes $Y \leftarrow \mathcal{F}_O(K_s, X)$ and $\Lambda_O \leftarrow L_O(K_s, X)$, and sends $(Y, (\Lambda_O, \Lambda_I))$ back to $\mathcal{A}_{\mathsf{lprf}}$. When $\mathcal{A}_{\mathsf{lprf}}$ outputs b', $\overline{\mathcal{A}}_{\mathsf{lprf}}$ outputs b'.

It is easy to see that $\overline{\mathcal{A}}_{\mathsf{lprf}}$ perfectly simulates G_0 or G_1 for $\mathcal{A}_{\mathsf{lprf}}$, depending on its own challenge. Also, all queries by $\overline{\mathcal{A}}_{\mathsf{lprf}}$ are permitted as it queries exactly the same values as $\mathcal{A}_{\mathsf{lprf}}$ and they play the same game. Hence we conclude with

$$\begin{aligned}
\left|\Pr[\mathcal{A}^{\mathsf{G}_0}_{\mathsf{lprf}} \Rightarrow 1] - \Pr[\mathcal{A}^{\mathsf{G}_1}_{\mathsf{lprf}} \Rightarrow 1]\right| &= \left|\Pr[\overline{\mathcal{A}}^{\mathsf{LPRF}}_{\mathsf{lprf}} \Rightarrow 1 \mid b = 1] - \Pr[\overline{\mathcal{A}}^{\mathsf{LPRF}}_{\mathsf{lprf}} \Rightarrow 1 \mid b = 0]\right| \\
&= \mathbf{Adv}^{\mathsf{LPRF}}_{\mathcal{F}_I}(\overline{\mathcal{A}}_{\mathsf{lprf}}, \mathcal{L}_I)\,. \qquad (3.66)
\end{aligned}$$

To bound the game hop between G_1 and G_2 we introduce a sequence of hybrid games $\mathsf{H}_0, \ldots, \mathsf{H}_q$, which are displayed in Figure 3.24. In H_0 all queries to F are answered using $\mathcal{F}_O$ on a randomly chosen key per query, hence it holds that $\mathsf{H}_0 = \mathsf{G}_1$. Likewise, in H_q all queries to F are answered by returning random values which yields $\mathsf{H}_q = \mathsf{G}_2$. It holds that

$$\begin{aligned}
\left|\Pr[\mathcal{A}^{\mathsf{G}_1}_{\mathsf{lprf}} \Rightarrow 1] - \Pr[\mathcal{A}^{\mathsf{G}_2}_{\mathsf{lprf}} \Rightarrow 1]\right| &= \left|\Pr[\mathcal{A}^{\mathsf{H}_0}_{\mathsf{lprf}} \Rightarrow 1] - \Pr[\mathcal{A}^{\mathsf{H}_q}_{\mathsf{lprf}} \Rightarrow 1]\right| \\
&\leq \sum_{i=1}^{q} \left|\Pr[\mathcal{A}^{\mathsf{H}_{i-1}}_{\mathsf{lprf}} \Rightarrow 1] - \Pr[\mathcal{A}^{\mathsf{H}_i}_{\mathsf{lprf}} \Rightarrow 1]\right|\,.
\end{aligned}$$

For the difference between H_{i-1} and H_i, we construct adversary $\overline{\mathcal{A}}_{\mathsf{i}}$ as follows. At the start, $\overline{\mathcal{A}}_{\mathsf{i}}$ samples a random master key K for $\mathcal{F}_I$, which it will use for the leakage queries by $\mathcal{A}_{\mathsf{lprf}}$. Whenever $\mathcal{A}_{\mathsf{lprf}}$ queries $(X, (L_O, L_I))$ to its leakage oracle, $\overline{\mathcal{A}}_{\mathsf{i}}$ (locally) computes $K_s \leftarrow \mathcal{F}_I(K, X)$, $\Lambda_I \leftarrow L_I(K, X)$, $Y \leftarrow \mathcal{F}_O(K_s, X)$, and $\Lambda_O \leftarrow L_O(K_s, X)$, and sends $(Y, (\Lambda_O, \Lambda_I))$ to $\mathcal{A}_{\mathsf{lprf}}$. Let $X_1, \ldots, X_q$ denote the queries that $\mathcal{A}_{\mathsf{lprf}}$ makes to F (which we assume to be distinct without loss of generality). The first $i-1$ queries $X_1, \ldots, X_{i-1}$ are answered by returning a randomly chosen bit string of length t, while the last $q-i$ queries are answered with $Y_i \leftarrow \mathcal{F}_O(K_s, X_i)$, where K_s is a key chosen at random for each of these queries. The i^{th} query X_i is forwarded by $\overline{\mathcal{A}}_{\mathsf{i}}$ to its own oracle F as is the response back to $\mathcal{A}_{\mathsf{lprf}}$. When $\mathcal{A}_{\mathsf{lprf}}$ eventually outputs a bit b', $\overline{\mathcal{A}}_{\mathsf{i}}$ outputs $1-b'$.

It holds that $\overline{\mathcal{A}}_{\mathsf{i}}$ perfectly simulates H_{i-1} and H_i for $\mathcal{A}_{\mathsf{lprf}}$ if its own challenge bit from game PRF is 1 and 0, respectively. Outputting $1-b'$ then yields

$$\begin{aligned}\left|\Pr[\mathcal{A}_{\mathsf{lprf}}^{\mathsf{H}_{i-1}} \Rightarrow 1] - \Pr[\mathcal{A}_{\mathsf{lprf}}^{\mathsf{H}_i} \Rightarrow 1]\right| &= \left|\Pr[\overline{\mathcal{A}}_{\mathsf{i}}^{\mathsf{PRF}} \Rightarrow 0 \,|\, b=0] - \Pr[\overline{\mathcal{A}}_{\mathsf{i}}^{\mathsf{PRF}} \Rightarrow 0 \,|\, b=1]\right| \\ &= \mathbf{Adv}_{\mathcal{F}_O}^{\mathsf{PRF}}(\overline{\mathcal{A}}_{\mathsf{i}})\,.\end{aligned}$$

By a standard hybrid argument [FM21, MF21], we define $\overline{\mathcal{A}}_{\mathsf{prf}}$ to be the adversary that picks $i \leftarrow\!\!\$\, [q]$ and then behaves like $\overline{\mathcal{A}}_{\mathsf{i}}$ described above, and obtain

$$\begin{aligned}\left|\Pr[\mathcal{A}_{\mathsf{lprf}}^{\mathsf{G}_1} \Rightarrow 1] - \Pr[\mathcal{A}_{\mathsf{lprf}}^{\mathsf{G}_2} \Rightarrow 1]\right| &\le \sum_{i=1}^{q} \left|\Pr[\mathcal{A}_{\mathsf{lprf}}^{\mathsf{H}_{i-1}} \Rightarrow 1] - \Pr[\mathcal{A}_{\mathsf{lprf}}^{\mathsf{H}_i} \Rightarrow 1]\right| \\ &\le q\,\mathbf{Adv}_{\mathcal{F}_O}^{\mathsf{PRF}}(\overline{\mathcal{A}}_{\mathsf{prf}})\,. \qquad (3.67)\end{aligned}$$

Inserting (3.66) and (3.67) in (3.65) yields

$$\begin{aligned}\mathbf{Adv}_{\mathcal{F}}^{\mathsf{LPRF}}(\mathcal{A}_{\mathsf{lprf}}, \mathcal{L}_F) &= \left|\Pr[\mathcal{A}_{\mathsf{lprf}}^{\mathsf{G}_0} \Rightarrow 1] - \Pr[\mathcal{A}_{\mathsf{lprf}}^{\mathsf{G}_2} \Rightarrow 1]\right| \\ &\le \mathbf{Adv}_{\mathcal{F}_I}^{\mathsf{LPRF}}(\overline{\mathcal{A}}_{\mathsf{lprf}}, \mathcal{L}_I) + q\,\mathbf{Adv}_{\mathcal{F}_O}^{\mathsf{PRF}}(\overline{\mathcal{A}}_{\mathsf{prf}})\,.\end{aligned}$$

We construct the following adversary $\mathcal{A}_{\mathsf{luf}}$ against $\mathcal{F}$. It picks X at random and queries $(X, (L_O, L_I))$ to Lkg to obtain (Λ_O, Λ_I). Recall that L_O simply outputs its first input. This yields $\Lambda_O = L_O(\mathcal{F}_I(K, X), X) = L_O(K_s, X) = K_s$, i.e., $\mathcal{A}_{\mathsf{luf}}$ learns the session key. Then $\mathcal{A}_{\mathsf{luf}}$ locally computes $Y \leftarrow \mathcal{F}_O(K_s, X)$ and queries (X, Y) to Guess which will set the flag win to true as $Y = \mathcal{F}_O(K_s, X) = \mathcal{F}_O(\mathcal{F}_I(K, X), X) = \mathcal{F}(K, X)$ and $\mathcal{A}_{\mathsf{luf}}$ did not query X to F, in fact, it did not query any input to F, hence $\mathcal{S} = \emptyset$. We thus conclude with

$$\mathbf{Adv}_{\mathcal{F}}^{\mathsf{LUF}}(\mathcal{A}_{\mathsf{luf}}, \mathcal{L}_F) = 1\,.$$

Collecting everything above proves the claim. □

Our LUF adversary exploits the fact that the game LUF allows to obtain leakage for the input for which the output is predicted. Hence, the adversary leaks the session key K_s for some input X, which enables it to perfectly predict the output. Note that this does not enable the adversary to predict an output for a different input. Our LUF adversary bypasses the LUF security of the inner function $\mathcal{F}_I$ and attacks the outer function $\mathcal{F}_O$

Games $\mathsf{G}_0, \mathsf{G}_1, \mathsf{G}_2$

$b \leftarrow_\$ \{0,1\}$
$K \leftarrow_\$ \mathcal{K}$
$b' \leftarrow \mathcal{A}^{\mathsf{F},\mathsf{LF}}_{\mathsf{lprf}}()$

oracle $\mathsf{F}(X)$ in G_0

$K_s \leftarrow \mathcal{F}_I(K, X)$
return $Y \leftarrow \mathcal{F}_O(K_s, X)$

oracle $\mathsf{F}(X)$ in G_1

$K_s \leftarrow_\$ \{0,1\}^s$
return $Y \leftarrow \mathcal{F}_O(K_s, X)$

oracle $\mathsf{F}(X)$ in G_2

return $Y \leftarrow_\$ \{0,1\}^t$

oracle $\mathsf{LF}(X, (L_O, L_I))$

$K_s \leftarrow \mathcal{F}_I(K, X)$
$\Lambda_I \leftarrow L_I(K, X)$
$Y \leftarrow \mathcal{F}_O(K_s, X)$
$\Lambda_O \leftarrow L_O(K_s, X)$
return $(Y, (\Lambda_O, \Lambda_I))$

Figure 3.23: Games G_0, G_1, and G_2 used in the proof of Lemma 3.3.4. The games share the leakage oracle LF, while each game uses its own challenge oracle as described.

Games H_i

$K \leftarrow_\$ \mathcal{K}$
$c \leftarrow 0$
$b' \leftarrow \mathcal{A}^{\mathsf{F},\mathsf{LF}}_{\mathsf{lprf}}()$

oracle $\mathsf{F}(X)$ in H_i

$c \leftarrow c + 1$
if $c \leq i$
  $Y \leftarrow_\$ \{0,1\}^t$
else
  $K_s \leftarrow_\$ \{0,1\}^s$
  $Y \leftarrow \mathcal{F}_O(K_s, X)$
return Y

oracle $\mathsf{LF}(X, (L_O, L_I))$

$K_s \leftarrow \mathcal{F}_I(K, X)$
$\Lambda_I \leftarrow L_I(K, X)$
$Y \leftarrow \mathcal{F}_O(K_s, X)$
$\Lambda_O \leftarrow L_O(K_s, X)$
return $(Y, (\Lambda_O, \Lambda_I))$

Figure 3.24: Hybrid games H_i used in the proof of Lemma 3.3.4. The games share the leakage oracles LEnc and LDec and differ only in the oracle Enc.

instead. Regarding the LPRF security, observe the following. The LPRF adversary is also able to obtain a session key K_s through its leakage oracle. However, this session key is only valid for the queried input and the game LPRF does not allow to query this input to the challenge oracle, which would make the notion unachievable anyway. We essentially show that it is sufficient to secure the master key K by deriving the session keys using an inner function with strong security guarantees (LPRF and LUF in our case).

This result is of independent interest as it shows that extra caution is judicious in the leakage setting. Our constructions show how well-known and established results from the leakage-free setting, like the equivalence between pseudorandomness and unpredictability [NR98], do not necessarily remain valid in the leakage setting.

3.3.2 Leakage Resilience of the N2 Construction

The N2 composition theorem by Barwell et al. [BMOS17] (cf. Theorem 2.3.6) is formulated from arbitrary symmetric encryption schemes and MACs, in particular, schemes where decryption/verification leakage might be different than leakage resulting from encryption/tagging. However, the FGHF′ construction exhibits a very specific structure. Note, from the previous theorems towards the FGHF′ composition theorem, that the leakage for encryption and decryption as well as tagging and verification is always the same, which follows from the specific structure. More specifically, the MAC $\textsc{Mac}[\mathcal{H}, \mathcal{F}']$ is a canonical MAC using the function that results from the composition of $\mathcal{H}$ and $\mathcal{F}'$, as such verification works just as tagging, except for the final comparison which we assume to be leakage-free. Similar, the encryption scheme SLENC is completely defined by its encryption algorithm which is an involution, in particular, $\mathtt{Enc}[\mathcal{F}, \mathcal{G}](K, N, \cdot) = \mathtt{Dec}[\mathcal{F}, \mathcal{G}](K, N, \cdot)$. We call such encryption schemes mirror-like encryption schemes. In this section, we recast the N2 composition to the special case of mirror-like encryption schemes and canonical MACs.

Unforgeability of Canonical MACs

For canonical MACs, we can reformulate SUF-CMLA security (cf. Definition 2.3.4) to the security notion SUF-CMLA$_c$ defined below, the subscript "c" indicating "canonical". The mere difference is that SUF-CMLA$_c$ does not provide a leakage verification oracle LVfy to the adversary as any information the adversary obtains from this is already captured by its leakage tagging oracle LTag, again assuming that the final tag comparison is leakage-free.

Definition 3.3.5 (SUF-CMLA$_c$ Security)**.** *Let* $\textsc{Mac} = (\mathtt{Tag}, \mathtt{Ver})$ *be a canonical message authentication code and the game* $\mathsf{SUFCMLA}_c$ *be as defined in Figure 3.25. For any adversary* $\mathcal{A}$ *that never forwards queries to or from the oracle* Vfy*, and only queries leakage functions to its oracle* LTag *in the respective set* $\mathcal{L}_T$*, its corresponding* SUF-CMLA$_c$ *advantage is given by*

$$\mathbf{Adv}^{\mathsf{SUFCMLA}_c}_{\textsc{Mac}}(\mathcal{A}, \mathcal{L}_T) \coloneqq \left|2\Pr[\mathsf{SUFCMLA}_c^{\mathcal{A}} \Rightarrow 1] - 1\right| .$$

Security of Mirror-Like Encryption

Since encryption and decryption leakage for mirror-like encryption schemes is the same, one can simply simulate decryption leakage using encryption leakage. For nonce-based

Game $\mathsf{SUFCMLA}_c$	**oracle** $\mathsf{Vfy}(X,T)$	**oracle** $\mathsf{LTag}(X,L)$
$b \leftarrow\!\!\$\ \{0,1\}$	**if** $b = 0$	$\Lambda \leftarrow L(K,X)$
$K \leftarrow\!\!\$\ \mathcal{K}$	**return** $\perp$	$T \leftarrow \mathtt{Tag}(K,X)$
$b' \leftarrow \mathcal{A}^{\mathsf{Vfy},\mathsf{LTag}}()$	**else**	**return** (T,Λ)
return $(b' = b)$	$V \leftarrow \mathtt{Ver}(K,X,T)$	
	return V	

Figure 3.25: Security game $\mathsf{SUFCMLA}_c$.

encryption schemes, however, the adversary is typically restricted to be nonce-respecting, meaning that it never repeats a nonce. But this restriction only ever applies to encryption oracles, modelling careful nonce-selection on the side of the encrypting party, while the adversary is free to arbitrarily repeat nonces to the decryption oracle. Simulating decryption leakage using encryption leakage now comes with the problem that the adversary is not nonce-respecting any more. But since we are only talking about leakage oracles here, we can still force the adversary to be nonce-respecting to the challenge oracles, or more precisely, the challenge encryption oracle. This is what we call semi-nonce-respecting adversaries. Precisely, these are adversaries which are nonce-respecting, i.e., they never repeat a nonce to the challenge encryption oracle but not to the leakage encryption oracle. This follows the definition of misuse-resilience given in [ADL17] and used for instance in [GPPS19].

Recast N2 Composition Theorem

We now recast the N2 composition theorem by Barwell et al. [BMOS17] (cf. Theorem 2.3.6) to the special case of mirror-like encryption schemes and canonical MACs. The theorem below shows that we can reduce the security of the resulting AEAD scheme to the security of the underlying canonical MAC and mirror-like encryption scheme. More precisely, LAE security reduces to the SUF-CMLA$_c$ and LPRF security of the underlying canonical MAC and to the IND-CPLA security for semi-nonce-respecting adversaries of the underlying mirror-like encryption scheme. In contrast, the theorem by Barwell et al. [BMOS17, Theorem 1] (cf. Theorem 2.3.6) reduces it to the SUF-CMLA and LPRF security of the MAC and IND-aCPLA against nonce-respecting adversaries of the encryption scheme. We first give the full proof for our setting. Subsequently, we discuss the difference between our proof and the one given in [BMOS17].

Theorem 3.3.6 (LAE Security of the N2 Construction)**.** *Let* $\textsc{Se} = (\mathtt{Enc},\mathtt{Dec})$ *be a mirror-like symmetric encryption scheme and* $\textsc{Mac} = (\mathtt{Tag},\mathtt{Ver})$ *be a canonical MAC with associated leakage sets* $\mathcal{L}_E$ *and* $\mathcal{L}_T$*, respectively. Let* N2 *be the composition of* Se *and* Mac *as displayed in Figure 3.1 with associated leakage sets* $\mathcal{L}_{AE} = \mathcal{L}_{VD} = \mathcal{L}_E \times \mathcal{L}_T$*. For any nonce-respecting adversary* $\mathcal{A}_{\mathsf{ae}}$ *against* N2 *there exists a semi-nonce-respecting* IND-CPLA *adversary* $\mathcal{A}_{\mathsf{se}}$ *against* Se*, an* LPRF *adversary* $\mathcal{A}_{\mathsf{lprf}}$ *against* Tag*, and a* SUF-CMLA$_c$ *adver-*

sary $\mathcal{A}_{\mathsf{mac}}$ *against* MAC *such that*

$$\mathbf{Adv}^{\mathsf{LAE}}_{\mathrm{N2}}(\mathcal{A}_{\mathsf{ae}}, \mathcal{L}_{AE}, \mathcal{L}_{VD}) \leq \mathbf{Adv}^{\mathsf{INDCPLA}}_{\mathrm{SE}}(\mathcal{A}_{\mathsf{se}}, \mathcal{L}_{E}) + \mathbf{Adv}^{\mathsf{LPRF}}_{\mathsf{Tag}}(\mathcal{A}_{\mathsf{lprf}}, \mathcal{L}_{T}) + \mathbf{Adv}^{\mathsf{SUFCMLA}_c}_{\mathrm{MAC}}(\mathcal{A}_{\mathsf{mac}}, \mathcal{L}_{T}) .$$

Proof. We us a sequence of games $\mathsf{G}_0, \ldots, \mathsf{G}_3$ shown in Figure 3.26. Game G_0 is the game LAE instantiated with N2 with secret bit fixed to 1. In game G_1, the decryption oracle is changed to reject any queried ciphertext. Game G_2 and G_3 are like G_1, except for the following differences. Game G_2 generates the challenge ciphertext by sampling it at random but still computing the real tag for this ciphertext. In game G_3 both the ciphertext and the tag are sampled at random. Hence G_3 equals the game LAE instantiated with N2 with secret bit fixed to 0. Using a simple reformulation to the adversarial advantage yields

$$\begin{aligned}
\mathbf{Adv}^{\mathsf{LAE}}_{\mathrm{N2}}(\mathcal{A}_{\mathsf{ae}}, \mathcal{L}_{AE}, \mathcal{L}_{VD}) &= \left|\Pr[\mathcal{A}^{\mathsf{LAE}}_{\mathsf{ae}} \Rightarrow 1 \mid b = 1] - \Pr[\mathcal{A}^{\mathsf{LAE}}_{\mathsf{ae}} \Rightarrow 1 \mid b = 0]\right| \\
&= \left|\Pr[\mathcal{A}^{\mathsf{G}_0}_{\mathsf{ae}} \Rightarrow 1] - \Pr[\mathcal{A}^{\mathsf{G}_3}_{\mathsf{ae}} \Rightarrow 1]\right| \\
&\leq \sum_{i=1}^{3} \left|\Pr[\mathcal{A}^{\mathsf{G}_{i-1}}_{\mathsf{ae}} \Rightarrow 1] - \Pr[\mathcal{A}^{\mathsf{G}_i}_{\mathsf{ae}} \Rightarrow 1]\right| . \qquad (3.68)
\end{aligned}$$

Below, we bound each of the game hops. The game hop from G_0 to G_1 is bound by the SUF-CMLA$_c$ security of the underlying MAC MAC, the game hop from G_1 to G_2 by the IND-CPLA security of the underlying encryption scheme SE, and the game hop from G_2 to G_3 by the LPRF security of the tagging algorithm $\mathtt{Tag}$.

We start with the adversarial advantage between games G_0 and G_1. We construct the following SUF-CMLA$_c$ adversary $\mathcal{A}_{\mathsf{mac}}$. It samples a random key K_e for the encryption scheme SE. Queries (N, A, M) to Enc are answered by $\mathcal{A}_{\mathsf{mac}}$ as follows. It computes the ciphertext $C_e \leftarrow \mathtt{Enc}(K_e, N, M)$, obtains the tag T by invoking its oracle LTag on $((N, A, C_e), \emptyset)$[17], and sends the ciphertext $C \leftarrow (C_e, T)$ back to $\mathcal{A}_{\mathsf{ae}}$. Leakage encryption queries $(N, A, M, (L_E, L_T))$ are processed as follows. The ciphertext $C_e \leftarrow \mathtt{Enc}(K_e, N, M)$ and corresponding leakage $\Lambda_E \leftarrow L_E(K_e, N, M)$ are computed locally by $\mathcal{A}_{\mathsf{mac}}$. Subsequently, it queries its leakage oracle LTag on $((N, A, C_e), L_T)$ to obtain (T, Λ_T) and sends back $C \leftarrow (C_e, T)$ and $\Lambda \leftarrow (\Lambda_E, \Lambda_T)$ to $\mathcal{A}_{\mathsf{ae}}$. For any leakage decryption query $(N, A, (C_e, T), (L_D, L_V))$, $\mathcal{A}_{\mathsf{mac}}$ queries its own leakage oracle LTag on $((N, A, C_e), L_V)$ to obtain T'. If $T' \neq T$, it returns $(\bot, \Lambda_V)$ to $\mathcal{A}_{\mathsf{ae}}$. Otherwise, i.e., $T' = T$, $\mathcal{A}_{\mathsf{mac}}$ computes $M \leftarrow \mathtt{Dec}(K_e, N, C_e)$ and $\Lambda_D \leftarrow L_E(K_e, N, C_e)$, and sends back $(M, \Lambda_D, \Lambda_V)$ to $\mathcal{A}_{\mathsf{ae}}$. Whenever $\mathcal{A}_{\mathsf{ae}}$ queries its oracle Dec on $(N, A, (C_e, T))$, $\mathcal{A}_{\mathsf{mac}}$ forwards $((N, A, C_e), T)$ to its oracle Vfy. If the response of Vfy is $\bot$, $\mathcal{A}_{\mathsf{mac}}$ forwards the response to $\mathcal{A}_{\mathsf{ae}}$, otherwise, $\mathcal{A}_{\mathsf{mac}}$ computes $M \leftarrow \mathtt{Dec}(K_e, N, C_e)$ and sends M to $\mathcal{A}_{\mathsf{ae}}$.

Clearly, $\mathcal{A}_{\mathsf{mac}}$ queries its leakage oracle LTag only on the permissive functions, as $\mathcal{A}_{\mathsf{ae}}$ does. Also, $\mathcal{A}_{\mathsf{mac}}$ does not make any prohibited query, as it invokes its challenge oracle Vfy if and only if $\mathcal{A}_{\mathsf{ae}}$ makes a query to its challenge decryption oracle Dec which never forwards any query to or from it.

Recall that the difference between G_0 and G_1 is that the former implements the real decryption oracle while the latter rejects any decryption query. Conditioned on the secret

[17] $\mathcal{A}_{\mathsf{mac}}$ does not submit a leakage function, as it simulates a challenge oracle for $\mathcal{A}_{\mathsf{ae}}$.

bit b of $\mathsf{SUFCMLA}_c$ being 0, $\mathcal{A}_{\mathsf{mac}}$ never decrypts C_e, hence it perfectly simulates G_1 for $\mathcal{A}_{\mathsf{ae}}$. Likewise, if $b = 1$, $\mathcal{A}_{\mathsf{mac}}$ only decrypts if the tag T is valid, thus it perfectly simulates G_0 for $\mathcal{A}_{\mathsf{ae}}$. Hence, we conclude with

$$\begin{aligned}\left|\Pr[\mathcal{A}_{\mathsf{ae}}^{\mathsf{G}_0} \Rightarrow 1] - \Pr[\mathcal{A}_{\mathsf{ae}}^{\mathsf{G}_1} \Rightarrow 1]\right| &= \left|\Pr[\mathcal{A}_{\mathsf{mac}}^{\mathsf{SUFCMLA}_c} \Rightarrow 1 \mid b = 1] - \Pr[\mathcal{A}_{\mathsf{mac}}^{\mathsf{SUFCMLA}_c} \Rightarrow 1 \mid b = 0]\right| \\ &= \mathbf{Adv}_{\mathrm{MAC}}^{\mathsf{SUFCMLA}_c}(\mathcal{A}_{\mathsf{mac}}, \mathcal{L}_T)\,. \end{aligned} \tag{3.69}$$

For the remaining game hops, note that the oracle Dec rejects any ciphertext irrespective of the validity of the tag, which is why we omit it in the description as every reduction simply responds with $\perp$.

For the game hop between G_1 and G_2, we construct an IND-CPLA adversary $\mathcal{A}_{\mathsf{se}}$ as follows. It generates a key K_m for MAC and runs the adversary $\mathcal{A}_{\mathsf{ae}}$ answering its queries as follows. For leakage queries $(N, A, M, (L_E, L_T))$ to LEnc it passes (N, M, L_E) to its own oracle LEnc and obtains (C_e, Λ_E) in return. It computes the tag $T \leftarrow \mathtt{Tag}(K_m, N, A, C_e)$, corresponding leakage $\Lambda_T \leftarrow L_T(K_m, N, A, C_e)$, and sends $((C_e, T), (\Lambda_E, \Lambda_T))$ back to $\mathcal{A}_{\mathsf{ae}}$. For leakage queries $(N, A, (C_e, T), (L_D, L_V))$ to LDec, the reduction $\mathcal{A}_{\mathsf{se}}$ computes $V \leftarrow \mathtt{Ver}(K_m, (N, A, C_e), T)$ and $\Lambda_V \leftarrow L_V(K_m, N, A, C_e, T)$. If $V = \perp$, it sends $(\perp, \Lambda_V)$ back to $\mathcal{A}_{\mathsf{ae}}$. If $V = \top$, it forwards (N, M, L_E) to its own leakage encryption oracle LEnc to obtain (M, Λ_D) and sends $(M, (\Lambda_D, \Lambda_V))$ back to $\mathcal{A}_{\mathsf{ae}}$. Note that $\mathtt{Enc}(K, N, C) = \mathtt{Dec}(K, N, C)$ since SE is a mirror-like encryption scheme. Queries (N, A, M) to Enc are handled by obtaining C_e from its own challenge encryption oracle invoked with (N, M), computing the tag $T \leftarrow \mathtt{Tag}(K_m, N, A, C_e)$, and sending (C_e, T) back to $\mathcal{A}_{\mathsf{ae}}$.

Since $\mathcal{A}_{\mathsf{ae}}$ queries its leakage oracles only on functions in the corresponding leakage set, so does $\mathcal{A}_{\mathsf{se}}$. Every challenge encryption query by $\mathcal{A}_{\mathsf{ae}}$ entails that $\mathcal{A}_{\mathsf{se}}$ invokes its challenge encryption query. Likewise, every leakage query, either encryption or decryption, leads to a leakage encryption query by $\mathcal{A}_{\mathsf{se}}$. As a valid LAE adversary, $\mathcal{A}_{\mathsf{ae}}$ does not forward queries from challenge to leakage oracles or vice versa, as does $\mathcal{A}_{\mathsf{se}}$. Note further that $\mathcal{A}_{\mathsf{se}}$ is semi-nonce-respecting. This follows from $\mathcal{A}_{\mathsf{se}}$ simulating both leakage oracles of $\mathcal{A}_{\mathsf{ae}}$ using its leakage encryption oracle and $\mathcal{A}_{\mathsf{ae}}$ being nonce-respecting.

It is easy to see that $\mathcal{A}_{\mathsf{se}}$ perfectly simulates either G_1 or G_2 for $\mathcal{A}_{\mathsf{se}}$. The games differ in the ciphertext part C_e generated by Enc. In G_1 it is the encryption of the message M, while it is a random bit string in G_2. By setting C_e to the output of its own challenge oracle, $\mathcal{A}_{\mathsf{se}}$ simulates G_1 and G_2 for $\mathcal{A}_{\mathsf{ae}}$ conditioned on the secret bit b of the game $\mathsf{INDCPLA}$ being 1 and 0, respectively. It holds that

$$\begin{aligned}\left|\Pr[\mathcal{A}_{\mathsf{ae}}^{\mathsf{G}_1} \Rightarrow 1] - \Pr[\mathcal{A}_{\mathsf{ae}}^{\mathsf{G}_2} \Rightarrow 1]\right| &= \left|\Pr[\mathcal{A}_{\mathsf{se}}^{\mathsf{INDCPLA}} \Rightarrow 1 \mid b = 1] - \Pr[\mathcal{A}_{\mathsf{se}}^{\mathsf{INDCPLA}} \Rightarrow 1 \mid b = 0]\right| \\ &= \mathbf{Adv}_{\mathrm{SE}}^{\mathsf{INDCPLA}}(\mathcal{A}_{\mathsf{se}}, \mathcal{L}_E)\,. \end{aligned} \tag{3.70}$$

Finally, we construct the following LPRF adversary $\mathcal{A}_{\mathsf{lprf}}$ to bound the adversarial advantage between G_2 and G_3. It generates a key K_e for the underlying encryption scheme. Leakage encryption queries $(N, A, M, (L_E, L_T))$ are processed by locally computing $C_e \leftarrow \mathtt{Enc}(K_e, N, M)$ and $\Lambda_E \leftarrow L_E(K_e, N, M)$, invoking LF on $((N, A, C_e), L_T)$ in order to obtain (T, Λ_T), and sending $((C_e, T), (\Lambda_E, \Lambda_T))$ back to $\mathcal{A}_{\mathsf{ae}}$. For leakage decryption queries $(N, A, (C_e, T), (L_D, L_V))$, $\mathcal{A}_{\mathsf{lprf}}$ sends $((N, A, C_e), L_V)$ to its leakage oracle LF to obtain

(T', Λ_V). If $T \neq T'$, $\mathcal{A}_{\mathsf{lprf}}$ sends $(\bot, \Lambda_V)$ to $\mathcal{A}_{\mathsf{ae}}$. Otherwise, $\mathcal{A}_{\mathsf{lprf}}$ locally computes $M \leftarrow \mathtt{Dec}(K_e, N, C_e)$ and $\Lambda_D \leftarrow L_D(K_e, N, C_e)$, and sends $(M, (\Lambda_D, \Lambda_V))$ to $\mathcal{A}_{\mathsf{ae}}$. For queries (N, A, M) that $\mathcal{A}_{\mathsf{ae}}$ makes to its challenge encryption oracle Enc, $\mathcal{A}_{\mathsf{lprf}}$ samples a random bit string C_e of appropriate length, invokes its challenge oracle F on (N, A, C_e) to obtain T, and sends (C_e, T) back to $\mathcal{A}_{\mathsf{ae}}$.

Recall that the difference between G_2 and G_3 is how the tag T is generated. In G_2 it is the real tag computed on a random ciphertext, in G_3 it is a random bit string. By construction, $\mathcal{A}_{\mathsf{lprf}}$ perfectly simulates G_2 and G_3 if its own challenge bit b (from the game LPRF) is equal to 1 and 0, respectively.

Every challenge (leakage) query by $\mathcal{A}_{\mathsf{lprf}}$ stems from a challenge (leakage) query by $\mathcal{A}_{\mathsf{ae}}$. As $\mathcal{A}_{\mathsf{ae}}$ does not forward queries between its challenge and leakage oracles neither does $\mathcal{A}_{\mathsf{lprf}}$. Hence, we conclude that $\mathcal{A}_{\mathsf{lprf}}$ is a valid LPRF adversary against $\mathtt{Tag}$, which yields

$$\begin{aligned}\left|\Pr[\mathcal{A}_{\mathsf{ae}}^{\mathsf{G}_2} \Rightarrow 1] - \Pr[\mathcal{A}_{\mathsf{ae}}^{\mathsf{G}_3} \Rightarrow 1]\right| &= \left|\Pr[\mathcal{A}_{\mathsf{lprf}}^{\mathsf{LPRF}} \Rightarrow 1 \,|\, b = 1] - \Pr[\mathcal{A}_{\mathsf{lprf}}^{\mathsf{LPRF}} \Rightarrow 1 \,|\, b = 0]\right| \\ &= \mathbf{Adv}_{\mathsf{Tag}}^{\mathsf{LPRF}}(\mathcal{A}_{\mathsf{lprf}}, \mathcal{L}_T)\,. \end{aligned} \tag{3.71}$$

Inserting (3.69), (3.70), and (3.71) in (3.68) proves the statement. □

We will now discuss the differences between our proof and the one from [BMOS17]. In [BMOS17], the first game hop differs in that it also changes the leakage decryption oracle LDec. The change is made such that any leakage decryption query which is not forwarded from the leakage encryption oracle is rejected by returning $\bot$. In [BMOS17], this additional change is necessary in order to bound the second game hop by the IND-aCPLA security of the underlying encryption scheme. To detect the difference, the LAE adversary $\mathcal{A}_{\mathsf{ae}}$ has to submit a (fresh) valid ciphertext to LDec as an invalid ciphertext would be rejected anyway. This entails that $\mathcal{A}_{\mathsf{ae}}$ has generated a (fresh) valid tag for this ciphertext, which the reduction will use to distinguish whether its challenge oracle implements the verification algorithm or $\bot$. Since the leakage decryption oracle is simulated via the leakage verification oracle, the reduction has to forward this leakage query to its own challenge oracle to distinguish between the real and the ideal world. This is exactly the query which prevents building such a MAC from a function which is pseudorandom and ultimately led to the introduction of LUF security.

The next game hop is the same as in our proof, except that the leakage decryption oracle does not decrypt any fresh ciphertext due to the change in the first game hop. This restriction allows to bound the second game hop by the IND-aCPLA security of the underlying encryption scheme, as the only queries that cannot be answered with the oracle from the game $\mathsf{INDaCPLA}$, i.e., decryption of fresh ciphertexts, are answered with $\bot$. In our case of mirror-like encryption schemes, this issue does not arise if the scheme is secure with respect to semi-nonce-respecting adversaries in which case we only need IND-CPLA security as forwarded leakage decryption queries are answered like fresh queries.

Likewise, the third game hop is essentially the same, again only differing in the leakage decryption oracle. Since the LPRF adversary simulates the encryption-related part of the game locally, this difference is trivial.

Finally, Barwell et al. [BMOS17] have a fourth game hop. In this game hop, where the challenge oracles are already idealised, they merely revert the change of the leakage

Games $\mathsf{G}_0, \mathsf{G}_1, \mathsf{G}_2, \mathsf{G}_3$

$K_e, K_m \leftarrow_\$ \mathcal{K}$
$b' \leftarrow \mathcal{A}_{\mathsf{ae}}^{\mathsf{Enc},\mathsf{LEnc},\mathsf{Dec},\mathsf{LDec}}()$

oracle $\mathsf{LEnc}(N, A, M, L)$

parse L **as** (L_E, L_T)
$C_e \leftarrow \mathtt{Enc}(K_e, N, M)$
$T \leftarrow \mathtt{Tag}(K_m, N, A, C_e)$
$C \leftarrow (C_e, T)$
$\Lambda_E \leftarrow L_E(K_e, N, M)$
$\Lambda_T \leftarrow L_T(K_m, N, A, C_e)$
$\Lambda \leftarrow (\Lambda_E, \Lambda_T)$
return (C, Λ)

oracle $\mathsf{LDec}(N, A, C, L)$

parse L **as** (L_D, L_V)
parse C **as** (C_e, T)
$V \leftarrow \mathtt{Ver}(K_m, N, A, C_e, T)$
$\Lambda_V \leftarrow L_V(K_m, N, A, C_e, T)$
if $V = \bot$
  return $(\bot, \Lambda_V)$
$M \leftarrow \mathtt{Dec}(K_e, N, C_e)$
$\Lambda_D \leftarrow L_D(K_e, N, C_e)$
return $(M, \Lambda_D, \Lambda_V)$

oracle $\mathsf{Enc}(N, A, M)$ in $\mathsf{G}_0, \mathsf{G}_1$

$C_e \leftarrow \mathtt{Enc}(K_e, N, M)$
$T \leftarrow \mathtt{Tag}(K_m, N, A, C_e)$
return $C \leftarrow (C_e, T)$

oracle $\mathsf{Enc}(N, A, M)$ in G_2

$C_e \leftarrow_\$ \{0,1\}^\gamma$
$T \leftarrow \mathtt{Tag}(K_m, N, A, C_e)$
return $C \leftarrow (C_e, T)$

oracle $\mathsf{Enc}(N, A, M)$ in G_3

$C_e \leftarrow_\$ \{0,1\}^\gamma$
$T \leftarrow_\$ \{0,1\}^\tau$
return $C \leftarrow (C_e, T)$

oracle $\mathsf{Dec}(N, A, C)$ in G_0

parse C **as** (C_e, T)
$V \leftarrow \mathtt{Ver}(K_m, N, A, C_e, T)$
if $V = \bot$
  return $\bot$
return $M \leftarrow \mathtt{Dec}(K_e, N, C_e)$

oracle $\mathsf{Dec}(N, A, C)$ in $\mathsf{G}_1, \mathsf{G}_2, \mathsf{G}_3$

return $\bot$

Figure 3.26: Games $\mathsf{G}_0, \mathsf{G}_1, \mathsf{G}_2, \mathsf{G}_3$ used in the proof of Theorem 3.3.6. The oracles LEnc and LDec are identical across the games. Each game uses the oracles Enc and Dec as specified.

decryption oracle from the first game hop in order to end up with the idealised game, that is LAE with secret bit 0. Since we never change any leakage oracle throughout our proof, we do not need this additional game hop.

We emphasise that the N2 composition theorem by Barwell et al. [BMOS17] is perfectly fine and, in fact, more general than our version above. But the generality of the former posed an insurmountable barrier for constructing leakage-resilient MACs from leakage-resilient pseudorandomness which ultimately led to LUF security. By recasting the N2 composition theorem to the special case of the FGHF′ construction, we circumvent the insurmountable barrier by weakening the requirements towards the MAC as there is no forwarding from leakage to challenge oracles any more. Simultaneously, this comes at the cost of strengthening the requirements towards the symmetric encryption scheme, namely by requiring security against semi-nonce-respecting adversaries. However, as we will show below, the encryption scheme $\textsc{Se}[\mathcal{F},\mathcal{G}]$ underlying the FGHF′ construction—without any modification—already achieves security against semi-nonce-respecting adversaries.

3.3.3 IND-CPLA and SUF-CMLA$_c$ from LPRF

In this section we show that the encryption scheme and MAC underlying the FGHF′ construction achieve the security notions requested by the recast N2 composition theorem above.

Encryption

The theorem below shows that $\textsc{Se}[\mathcal{F},\mathcal{G}]$ achieves IND-CPLA security against semi-nonce-respecting adversaries if the underlying function $\mathcal{F}$ achieves LPRF security and the underlying pseudorandom generator $\mathcal{G}$ is a PRG. The proof works essentially as the proof of Theorem 3.1.1 but we state it nevertheless for completeness.

Theorem 3.3.7. *Let* $\textsc{Se}[\mathcal{F},\mathcal{G}]$ *be the encryption scheme depicted in Figure 3.1, composed of the function family* $\mathcal{F}\colon \mathcal{K}\times\{0,1\}^{\nu}\to\{0,1\}^{n}$ *and the pseudorandom generator* $\mathcal{G}\colon \{0,1\}^{n}\to\{0,1\}^{*}$ *with respective associated leakage sets* $\mathcal{L}_F$ *and* $\mathcal{L}_G$*. Then for any semi-nonce-respecting* IND-CPLA *adversary* $\mathcal{A}_{\mathsf{se}}$ *against* $\textsc{Se}[\mathcal{F},\mathcal{G}]$*, making* q *queries to* Enc*, and associated leakage sets* $\mathcal{L}_E=\mathcal{L}_F\times\mathcal{L}_G$*, there exist an* LPRF *adversary* $\mathcal{A}_{\mathsf{lprf}}$ *against* $\mathcal{F}$ *and a* PRG *adversary* $\mathcal{A}_{\mathsf{prg}}$ *against* $\mathcal{G}$ *such that*

$$\mathbf{Adv}^{\mathsf{INDCPLA}}_{\textsc{Se}[\mathcal{F},\mathcal{G}]}(\mathcal{A}_{\mathsf{se}},\mathcal{L}_E)\le\mathbf{Adv}^{\mathsf{LPRF}}_{\mathcal{F}}(\mathcal{A}_{\mathsf{lprf}},\mathcal{L}_F)+q\,\mathbf{Adv}^{\mathsf{PRG}}_{\mathcal{G}}(\mathcal{A}_{\mathsf{prg}})\,.$$

Proof. The proof uses games G_0,G_1,G_2, and G_3 displayed in Figure 3.27. Game G_0 is the game INDCPLA instantiated with $\textsc{Se}[\mathcal{F},\mathcal{G}]$ and secret bit fixed to 1 and game G_3 is the

same with secret bit fixed to 0. It holds that

$$\begin{aligned}\mathbf{Adv}^{\mathsf{INDCPLA}}_{\mathrm{SE}[\mathcal{F},\mathcal{G}]}(\mathcal{A}_{\mathsf{se}},\mathcal{L}_E) &= \left|2\Pr[\mathsf{INDCPLA}^{\mathcal{A}_{\mathsf{se}}}\Rightarrow 1]-1\right| \\ &= \left|\Pr[\mathcal{A}^{\mathsf{INDCPLA}}_{\mathsf{se}}\Rightarrow 1\,|\,b=1]-\Pr[\mathcal{A}^{\mathsf{INDCPLA}}_{\mathsf{se}}\Rightarrow 1\,|\,b=0]\right| \\ &= \left|\Pr[\mathcal{A}^{\mathsf{G}_0}_{\mathsf{se}}\Rightarrow 1]-\Pr[\mathcal{A}^{\mathsf{G}_3}_{\mathsf{se}}\Rightarrow 1]\right| \\ &\leq \sum_{i=1}^{3}\left|\Pr[\mathcal{A}^{\mathsf{G}_{i-1}}_{\mathsf{se}}\Rightarrow 1]-\Pr[\mathcal{A}^{\mathsf{G}_i}_{\mathsf{se}}\Rightarrow 1]\right|. \end{aligned} \tag{3.72}$$

To bound the adversarial advantage between G_0 and G_1 we construct the following LPRF adversary $\mathcal{A}_{\mathsf{lprf}}$ against $\mathcal{F}$. Queries (N,M) to Enc by $\mathcal{A}_{\mathsf{se}}$, are answered by forwarding N to the oracle F to obtain z. Subsequently, $\mathcal{A}_{\mathsf{lprf}}$ computes $C\leftarrow\mathcal{G}(z,|M|)\oplus M$ and sends it to $\mathcal{A}_{\mathsf{se}}$. For leakage queries $(N,M,(L_F,L_G))$, $\mathcal{A}_{\mathsf{lprf}}$ does the following. It obtains the tuple (z,Λ_F) from the oracle LF upon querying it on (N,L_F), while $C\leftarrow\mathcal{G}(z,|M|)\oplus M$ and $\Lambda_G\leftarrow L_G(z,M)$ are computed locally before sending $(C,(\Lambda_F,\Lambda_G))$ to $\mathcal{A}_{\mathsf{se}}$. The crucial part is that the adversary, since it is considered semi-nonce-respecting, can repeat nonces across its leakage queries. This results in repeating queries of $\mathcal{A}_{\mathsf{lprf}}$ to its leakage oracle LF. This is no hindrance as the game LPRF permits such queries, in case of non-adaptive leakage, the response will actually be identical, i.e., $\mathcal{A}_{\mathsf{lprf}}$ could also maintain a table to look up the values whenever $\mathcal{A}_{\mathsf{se}}$ repeats a nonce in a query to LEnc. When $\mathcal{A}$ outputs a bit b', $\mathcal{A}_{\mathsf{lprf}}$ forwards it as its own output.

Since the games G_0 and G_1 solely differ in generating the seed z for $\mathcal{G}$ during queries to Enc, $\mathcal{A}_{\mathsf{lprf}}$ perfectly simulates game G_0 and G_1 conditioned on the secret bit of the game LPRF being 1 and 0, respectively. We conclude with

$$\begin{aligned}\left|\Pr[\mathcal{A}^{\mathsf{G}_0}_{\mathsf{se}}\Rightarrow 1]-\Pr[\mathcal{A}^{\mathsf{G}_1}_{\mathsf{se}}\Rightarrow 1]\right| &= \left|\Pr[\mathcal{A}^{\mathsf{LPRF}}_{\mathsf{lprf}}\Rightarrow 1\,|\,b=1]-\Pr[\mathcal{A}^{\mathsf{LPRF}}_{\mathsf{lprf}}\Rightarrow 1\,|\,b=0]\right| \\ &= \mathbf{Adv}^{\mathsf{LPRF}}_{\mathcal{F}}(\mathcal{A}_{\mathsf{lprf}},\mathcal{L}_F). \end{aligned} \tag{3.73}$$

Next, we bound the adversarial advantage between G_1 and G_2. Therefore, we introduce a sequence of hybrid games $\mathsf{H}_0,\ldots,\mathsf{H}_q$, displayed in Figure 3.28, which differ in how the value Z is generated. In H_i, for the first i queries, it is a random bit string of length identical to the queried message. For the remaining $q-i$ queries, it is the output of the PRG $\mathcal{G}$ on input a random seed and the length of the queried message. It holds that $\mathsf{H}_0=\mathsf{G}_1$ and $\mathsf{H}_q=\mathsf{G}_2$, hence

$$\begin{aligned}\left|\Pr[\mathcal{A}^{\mathsf{G}_1}_{\mathsf{se}}\Rightarrow 1]-\Pr[\mathcal{A}^{\mathsf{G}_2}_{\mathsf{se}}\Rightarrow 1]\right| &= \left|\Pr[\mathcal{A}^{\mathsf{H}_0}_{\mathsf{se}}\Rightarrow 1]-\Pr[\mathcal{A}^{\mathsf{H}_q}_{\mathsf{se}}\Rightarrow 1]\right| \\ &\leq \sum_{i=1}^{q}\left|\Pr[\mathcal{A}^{\mathsf{H}_{i-1}}_{\mathsf{se}}\Rightarrow 1]-\Pr[\mathcal{A}^{\mathsf{H}_i}_{\mathsf{se}}\Rightarrow 1]\right|. \end{aligned}$$

We construct PRG adversaries $\mathcal{B}_1,\ldots,\mathcal{B}_q$ to bound the game hops between each pair of consecutive hybrid games. At the beginning, adversary $\mathcal{B}_i$ samples a key K for the function $\mathcal{F}$, which allows to perfectly simulate the leakage oracle LEnc for $\mathcal{A}_{\mathsf{se}}$. For leakage queries $(N,M,(L_F,L_G))$, it (locally) computes $z\leftarrow\mathcal{F}(K,N)$, $C\leftarrow\mathcal{G}(z,|M|)\oplus M$, $\Lambda_F\leftarrow$

$L_F(K,N)$, and $\Lambda_G \leftarrow L_G(z,M)$, and sends $(C,(\Lambda_F,\Lambda_G))$ to $\mathcal{A}_{\mathsf{se}}$. Denote the challenge queries that $\mathcal{A}_{\mathsf{se}}$ makes to Enc by $(N_1,M_1),\ldots,(N_q,M_q)$. For queries (N_j,M_j) with $j<i$, $\mathcal{B}_{\mathsf{i}}$ samples $Z \leftarrow\!\!\$\ \{0,1\}^{\mu}$ and sends $C \leftarrow Z \oplus M_j$ back to $\mathcal{A}_{\mathsf{se}}$. For queries (N_j,M_j) with $j>i$, $\mathcal{B}_{\mathsf{i}}$ samples a seed z for $\mathcal{G}$ at random, computes $Z \leftarrow \mathcal{G}(z,|M_j|)$ and sends $C \leftarrow Z \oplus M_j$ back to $\mathcal{A}_{\mathsf{se}}$. For the i^{th} query, (N_i,M_i), $\mathcal{B}_{\mathsf{i}}$ invokes its own oracle G on $|M_i|$ to obtain Z, computes $C \leftarrow Z \oplus M_i$, and sends C back to $\mathcal{A}_{\mathsf{se}}$. It follows that $\mathcal{B}_{\mathsf{i}}$ simulates H_{i-1} and H_i if its challenge bit b from the game PRG is 1 and 0, respectively. We get

$$\begin{aligned}\left|\Pr[\mathcal{A}_{\mathsf{se}}^{\mathsf{H}_{i-1}} \Rightarrow 1] - \Pr[\mathcal{A}_{\mathsf{se}}^{\mathsf{H}_i} \Rightarrow 1]\right| &= \left|\Pr[\mathcal{B}_{\mathsf{i}}^{\mathsf{PRG}} \Rightarrow 1 \mid b=1] - \Pr[\mathcal{B}_{\mathsf{i}}^{\mathsf{PRG}} \Rightarrow 1 \mid b=1]\right| \\ &= \mathbf{Adv}_{\mathcal{G}}^{\mathsf{PRG}}(\mathcal{B}_{\mathsf{i}})\,.\end{aligned}$$

By a standard hybrid argument [FM21, MF21], we define $\mathcal{A}_{\mathsf{prg}}$ to be the adversary that picks $i \leftarrow\!\!\$\ [q]$ and then behaves as $\mathcal{B}_{\mathsf{i}}$, and obtain

$$\begin{aligned}\left|\Pr[\mathcal{A}_{\mathsf{se}}^{\mathsf{G}_1} \Rightarrow 1] - \Pr[\mathcal{A}_{\mathsf{se}}^{\mathsf{G}_2} \Rightarrow 1]\right| &\le \sum_{i=1}^{q} \left|\Pr[\mathcal{A}_{\mathsf{se}}^{\mathsf{H}_{i-1}} \Rightarrow 1] - \Pr[\mathcal{A}_{\mathsf{se}}^{\mathsf{H}_i} \Rightarrow 1]\right| \\ &\le q\,\mathbf{Adv}_{\mathcal{G}}^{\mathsf{PRG}}(\mathcal{A}_{\mathsf{prg}})\,. \qquad (3.74)\end{aligned}$$

The adversarial advantage between G_2 and G_3 is easily seen to be 0 as the challenge oracle Enc outputs identically distributed ciphertexts, which yields

$$\left|\Pr[\mathcal{A}_{\mathsf{se}}^{\mathsf{G}_2} \Rightarrow 1] - \Pr[\mathcal{A}_{\mathsf{se}}^{\mathsf{G}_3} \Rightarrow 1]\right| = 0\,. \qquad (3.75)$$

Inserting (3.73), (3.74), and (3.75) into (3.72) yields

$$\mathbf{Adv}^{\mathsf{INDCPLA}}_{\mathrm{SE}[\mathcal{F},\mathcal{G}]}(\mathcal{A}_{\mathsf{se}},\mathcal{L}_E,\mathcal{L}_D) \le \mathbf{Adv}^{\mathsf{LPRF}}_{\mathcal{F}}(\mathcal{A}_{\mathsf{lprf}},\mathcal{L}_F) + q\,\mathbf{Adv}^{\mathsf{PRG}}_{\mathcal{G}}(\mathcal{A}_{\mathsf{prg}})\,,$$

which proves the theorem. □

MAC

The following theorem shows that we can construct a SUF-CMLA$_c$ secure canonical MAC from a function that is an LPRF. The proof differs substantially from the one of Theorem 3.1.3.

Theorem 3.3.8. *Let $\mathcal{F}\colon \mathcal{K}\times\mathcal{X}\to\{0,1\}^{\tau}$ be a function family with associated leakage set $\mathcal{L}_F$, and let $\textsc{Mac}[\mathcal{F}]$ be the corresponding canonical MAC with associated leakage sets $\mathcal{L}_T = \mathcal{L}_F$. Then, for any* SUF-CMLA$_c$ *adversary $\mathcal{A}_{\mathsf{mac}}$ against $\textsc{Mac}[\mathcal{F}]$ which makes q queries to Vfy, there exists an adversary $\mathcal{A}_{\mathsf{lprf}}$ against $\mathcal{F}$ such that*

$$\mathbf{Adv}^{\mathsf{SUFCMLA}_c}_{\textsc{Mac}[\mathcal{F}]}(\mathcal{A}_{\mathsf{mac}},\mathcal{L}_T) \le \mathbf{Adv}^{\mathsf{LPRF}}_{\mathcal{F}}(\mathcal{A}_{\mathsf{lprf}},\mathcal{L}_F) + \frac{q}{2^{\tau}-q}\,.$$

Proof. We prove the theorem using games G_0, G_1, and G_2 (cf. Figure 3.29), where G_0 and G_2 are the game $\mathsf{SUFCMLA}_c$ with secret bit 1 and 0, respectively. Using adversarial

Games $\mathsf{G}_0, \mathsf{G}_1, \mathsf{G}_2, \mathsf{G}_3$

$K \leftarrow_\$ \mathcal{K}$
$b' \leftarrow \mathcal{A}^{\mathsf{Enc},\mathsf{LEnc}}()$

oracle $\mathsf{Enc}(N, M)$ in G_0

$z \leftarrow \mathcal{F}(K, N)$
$Z \leftarrow \mathcal{G}(z, \mu)$
return $C \leftarrow Z \oplus M$

oracle $\mathsf{Enc}(N, M)$ in G_1

$z \leftarrow_\$ \{0,1\}^n$
$Z \leftarrow \mathcal{G}(z, \mu)$
return $C \leftarrow Z \oplus M$

oracle $\mathsf{Enc}(N, M)$ in G_2

$Z \leftarrow_\$ \{0,1\}^\mu$
return $C \leftarrow Z \oplus M$

oracle $\mathsf{Enc}(N, M)$ in G_3

return $C \leftarrow_\$ \{0,1\}^\mu$

oracle $\mathsf{LEnc}(N, M, (L_F, L_G))$

$z \leftarrow \mathcal{F}(K, N)$
$Z \leftarrow \mathcal{G}(z, \mu)$
$C \leftarrow Z \oplus M$
$\Lambda_F \leftarrow L_F(K, N)$
$\Lambda_G \leftarrow L_G(z, M)$
return $(C, (\Lambda_F, \Lambda_G))$

Figure 3.27: Games G_0, G_1, G_2, and G_3 used in the proof of Theorem 3.3.7. The oracle LEnc is shared across all games. Each game uses the oracle Enc as specified.

Game H_i

$K \leftarrow_\$ \mathcal{K}$
$c \leftarrow 0$
$b' \leftarrow \mathcal{A}^{\mathsf{Enc},\mathsf{LEnc}}()$

oracle $\mathsf{Enc}(N, M)$ in H_i

$c \leftarrow c + 1$
if $c \leq i$
  $Z \leftarrow_\$ \{0,1\}^{|M|}$
else
  $z \leftarrow_\$ \{0,1\}^n$
  $Z \leftarrow \mathcal{G}(z, |M|)$
return $C \leftarrow Z \oplus M$

oracle $\mathsf{LEnc}(N, M, (L_F, L_G))$

$z \leftarrow \mathcal{F}(K, N)$
$Z \leftarrow \mathcal{G}(z, |M|)$
$C \leftarrow Z \oplus M$
$\Lambda_F \leftarrow L_F(K, N)$
$\Lambda_G \leftarrow L_G(z, M)$
return $(C, (\Lambda_F, \Lambda_G))$

Figure 3.28: Hybrid games H_i used in the proof of Theorem 3.3.7. The games share the leakage oracle LEnc and differ only in the oracle Enc.

advantage, we obtain

$$\begin{aligned}\mathbf{Adv}^{\mathsf{SUFCMLA}_c}_{\textsc{Mac}[\mathcal{F}]}(\mathcal{A}_{\mathsf{mac}}), \mathcal{L}_T &= \left|2\ \Pr[\mathsf{SUFCMLA}_c^{\mathcal{A}_{\mathsf{mac}}} \Rightarrow 1] - 1\right| \\ &= \left|\Pr[\mathcal{A}_{\mathsf{mac}}^{\mathsf{SUFCMLA}_c} \Rightarrow 1 \,|\, b = 1] - \Pr[\mathcal{A}_{\mathsf{mac}}^{\mathsf{SUFCMLA}_c} \Rightarrow 1 \,|\, b = 0]\right| \\ &= \left|\Pr[\mathcal{A}_{\mathsf{mac}}^{\mathsf{G}_0} \Rightarrow 1] + \Pr[\mathcal{A}_{\mathsf{mac}}^{\mathsf{G}_2} \Rightarrow 1]\right| \\ &\leq \sum_{i=1}^{2} \left|\Pr[\mathcal{A}_{\mathsf{mac}}^{\mathsf{G}_{i-1}} \Rightarrow 1] - \Pr[\mathcal{A}_{\mathsf{mac}}^{\mathsf{G}_i} \Rightarrow 1]\right| . \end{aligned} \tag{3.76}$$

To bound the first term, we construct an LPRF adversary $\mathcal{A}_{\mathsf{lprf}}$ against $\mathcal{F}$ as follows. It runs $\mathcal{A}_{\mathsf{mac}}$ and answers leakage queries to LTag using its own leakage oracle LF. More formally, queries (X, L_T) to LTag are forwarded to LF as is the response (T, Λ_T) back to $\mathcal{A}_{\mathsf{mac}}$. Whenever $\mathcal{A}_{\mathsf{mac}}$ makes a query (X, T) to Vfy, $\mathcal{A}_{\mathsf{lprf}}$ forwards X to F to obtain T'. If $T = T'$ it sends $\top$ to $\mathcal{A}_{\mathsf{mac}}$, otherwise, it sends $\bot$.

It is easy to see that $\mathcal{A}_{\mathsf{lprf}}$ perfectly simulates G_0 and G_1 for $\mathcal{A}_{\mathsf{mac}}$ conditioned on the secret bit of game LPRF being 1 and 0, respectively. For every leakage (challenge) query by $\mathcal{A}_{\mathsf{mac}}$, $\mathcal{A}_{\mathsf{lprf}}$ makes exactly one leakage (challenge) query. As a SUF-CMLA$_c$ adversary, $\mathcal{A}_{\mathsf{mac}}$ does not forward queries to or from its challenge oracle, which yields that $\mathcal{A}_{\mathsf{lprf}}$ does not make any prohibited query. Hence we conclude with

$$\begin{aligned}\left|\Pr[\mathcal{A}_{\mathsf{mac}}^{\mathsf{G}_0} \Rightarrow 1] - \Pr[\mathcal{A}_{\mathsf{mac}}^{\mathsf{G}_1} \Rightarrow 1]\right| &= \left|\Pr[\mathcal{A}_{\mathsf{lprf}}^{\mathsf{LPRF}} \Rightarrow 1 \,|\, b = 1] - \Pr[\mathcal{A}_{\mathsf{lprf}}^{\mathsf{LPRF}} \Rightarrow 1 \,|\, b = 1]\right| \\ &= \mathbf{Adv}^{\mathsf{LPRF}}_{\mathcal{F}}(\mathcal{A}_{\mathsf{lprf}}, \mathcal{L}_F) . \end{aligned} \tag{3.77}$$

For the second term, note that the leakage oracle LTag are independent of the challenge oracle Vfy in both G_1 and G_2, thus we only consider the challenge oracle Vfy. To distinguish the games, the adversary $\mathcal{A}_{\mathsf{mac}}$ has to make a query (X, T) to Vfy, which would result in $\top$ in G_1. This means that $\mathcal{A}_{\mathsf{mac}}$ has to guess the value $f[X]$ for an arbitrary X. Since $f[\,]$ is sampled randomly from $\{0,1\}^\tau$, we have

$$\left|\Pr[\mathcal{A}_{\mathsf{mac}}^{\mathsf{G}_1} \Rightarrow 1] - \Pr[\mathcal{A}_{\mathsf{mac}}^{\mathsf{G}_2} \Rightarrow 1]\right| \leq \frac{q}{2^\tau - q} . \tag{3.78}$$

Inserting (3.77) and (3.78) into (3.76) proves the claim. □

Note that the above theorem states that for any LPRF $\mathcal{F}$ the canonical MAC $\textsc{Mac}[\mathcal{F}]$ is SUF-CMLA$_c$ secure with the same message space as $\mathcal{F}$. In order to let the MAC handle arbitrarily long inputs, we need $\mathcal{F}$ to handle arbitrarily long inputs. This is achieved by first hashing the (arbitrarily long) input using a collision-resistant hash function and then applying the function $\mathcal{F}$. The resulting construction yields an LPRF with arbitrary input length as stated in Theorem 3.1.6.

3.3.4 The Improved FGHF′ Composition Theorem

We can now state our improved composition theorem, which states that the FGHF′ construction yields an LAE-secure AEAD scheme, if the underlying functions $\mathcal{F}$ and $\mathcal{F}'$ are

Games $\mathsf{G}_0, \mathsf{G}_1, \mathsf{G}_2$

$K \leftarrow_{\$} \mathcal{K}$
$b' \leftarrow \mathcal{A}^{\mathsf{Vfy},\mathsf{LTag},\mathsf{LVfy}}()$

oracle $\mathsf{Vfy}(X, T)$ in G_0

$T' \leftarrow \mathcal{F}(K, X)$
return $(T' = T)$

oracle $\mathsf{Vfy}(X, T)$ in G_1

if $f[X] = \bot$
  $f[X] \leftarrow_{\$} \{0,1\}^{\tau}$
return $(f[X] = T)$

oracle $\mathsf{Vfy}(X, T)$ in G_2

return $\bot$

oracle $\mathsf{LTag}(X, L)$

$\Lambda \leftarrow L(K, X)$
$T \leftarrow \mathcal{F}(K, X)$
return (T, Λ)

Figure 3.29: Games $\mathsf{G}_0, \mathsf{G}_1, \mathsf{G}_2$ used in the proof of Theorem 3.3.8. The leakage oracle LTag is the same across the games. Each game uses the oracle Vfy as specified above.

leakage-resilient pseudorandom, $\mathcal{G}$ is a secure PRG, and $\mathcal{H}$ is a collision-resistant hash function. The theorem follows directly from Theorem 3.1.6, Theorem 3.3.6, Theorem 3.3.7, and Theorem 3.3.8. The implications are illustrated in Figure 3.30.

Theorem 3.3.9 (LAE Security of the FGHF′ Construction)**.** *Let $\mathcal{F}$ be a fixed-input-length function, $\mathcal{G}$ be a* PRG*, $\mathcal{H}$ be a vector hash function, and $\mathcal{F}'$ be a fixed-input-length function with associated leakage sets $\mathcal{L}_F$, $\mathcal{L}_G$, $\mathcal{L}_H$, and $\mathcal{L}_{F'}$, respectively. Let further $\mathcal{F}'$ output bit strings of length τ. Let* FGHF′ *be the composition of $\mathcal{F}$, $\mathcal{G}$, $\mathcal{H}$, and $\mathcal{F}'$ with associated leakage sets $\mathcal{L}_{AE} = \mathcal{L}_{VD} = \mathcal{L}_F \times \mathcal{L}_G \times \mathcal{L}_H \times \mathcal{L}_{F'}$, as depicted in Figure 3.1. Then for any nonce-respecting* LAE *adversary $\mathcal{A}_{\mathsf{ae}}$ against* FGHF′*, making q_E and q_D queries to* Enc *and* Dec*, respectively, there exist adversaries $\mathcal{A}_{\mathsf{lprf}}$, $\overline{\mathcal{A}}_{\mathsf{lprf}}$, $\mathcal{A}_{\mathsf{prg}}$, and $\mathcal{A}_{\mathsf{hash}}$ such that*

$$\mathbf{Adv}^{\mathsf{LAE}}_{\mathrm{FGHF}'}(\mathcal{A}_{\mathsf{ae}}, \mathcal{L}_{AE}, \mathcal{L}_{VD}) \leq \mathbf{Adv}^{\mathsf{LPRF}}_{\mathcal{F}}(\mathcal{A}_{\mathsf{lprf}}, \mathcal{L}_F) + 2\,\mathbf{Adv}^{\mathsf{LPRF}}_{\mathcal{F}'}(\overline{\mathcal{A}}_{\mathsf{lprf}}, \mathcal{L}_{F'}) + q_E\,\mathbf{Adv}^{\mathsf{PRG}}_{\mathcal{G}}(\mathcal{A}_{\mathsf{prg}}) + 2\,\mathbf{Adv}^{\mathsf{CR}}_{\mathcal{H}}(\mathcal{A}_{\mathsf{hash}}) + \frac{q_D}{2^{\tau} - q_D}\,.$$

Theorem 3.3.9 improves Theorem 3.1.7 by removing the additional requirement of unpredictability under leakage imposed on $\mathcal{F}'$. This entails that any instantiation of the FGHF′ construction can rely on the same function to instantiate $\mathcal{F}$ and $\mathcal{F}'$. Indeed, the sponge-based instantiation Slae uses the same function to instantiate $\mathcal{F}$ and $\mathcal{F}'$, however, pseudorandomness and unpredictability under leakage were proven separately.

3.4 Summary and Outlook

We developed the FGHF′ construction as a template for constructing non-adaptively leakage-resilient AEAD schemes from relatively simpler primitives—requiring only two

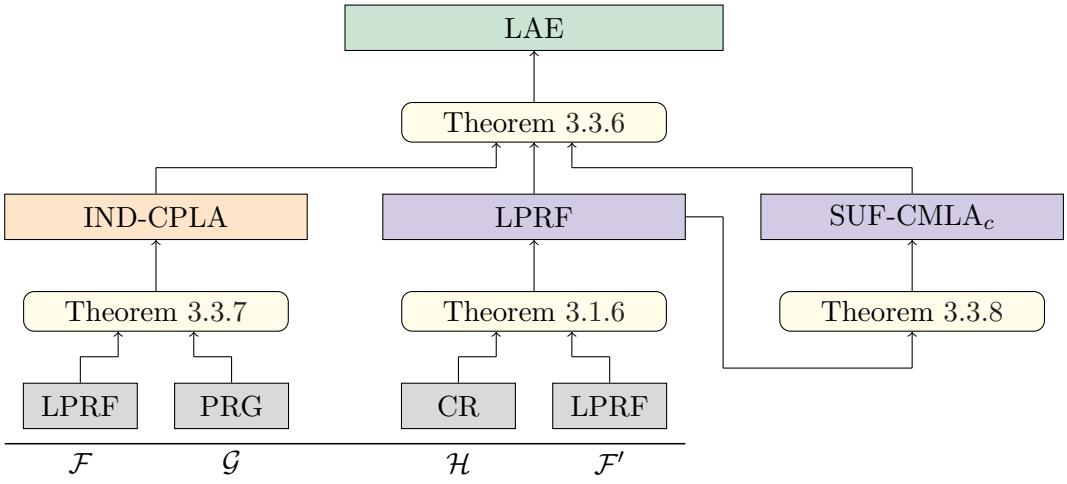

Figure 3.30: Visualisation of the FGHF′ composition theorem (cf. Theorem 3.3.9).

calls to the leakage-resilient functions per encryption or decryption call. We further presented Slae as a sponge-based instantiation of the FGHF′ construction, offering good performance and simplicity. Our security analysis shows that if the absorption rate is set sufficiently low, the transformation sponge yields a leakage-resilient function with the desired properties. However, some care is needed in interpreting these results. Like most treatments of leakage resilience, we assume that the leakage per evaluation is limited and does not drain the entropy in the secret state. Thus, it is implicitly assumed that an implementation is good enough to withstand basic side-channel attacks like Simple Power Analysis (SPA).

In the FGHF′ construction and Slae, authenticity is verified by recomputing the MAC tag and testing for equality between the recomputed tag and the one included in the ciphertext. While our leakage model accounts for the leakage that may take place during the tag recomputation, equality testing is assumed to be leak-free. In practice, however, this comparison might leak information [BBC+20]. Thus, any implementation of Slae (or any other realisation of the FGHF′ construction) needs to ensure that equality testing does not leak, or take additional measures, such as masking, to protect against leakage from this component. Dobraunig and Mennink [DM21] provide techniques on securing this value comparison against side-channel leakage.

A question that we did not address in this work, but nevertheless want to give an approach, is the instantiation of ρ. The security of Slae relies on it being instantiated with a non-invertible transformation rather than a permutation. On the other hand, most practical schemes employ permutations, such as Keccak [BDPA13] and Xoodoo [DHP+18]. A natural candidate to use for Slae is $\rho(x) = p(x) \oplus x$ for $p \in \{\text{Keccak}, \text{Xoodoo}\}$. Although this construction is known not to achieve indifferentiability [MRH04] from a random transformation, when given access to p, this should not preclude it from being a suitable candidate for instantiating constructions in the random transformation model. Indeed, Keccak and Xoodoo are also differentiable from a random permutation when given access to their underlying building blocks.

Finally, providing an implementation of SLAE which is hardened to bound the potential leakage an adversary can obtain is an interesting challenge. This effectively boils down to provide hardened implementations of the underlying function SLFUNC. The same applies for the generic FGHF′ construction for which an implementation for microcontrollers is given in [USS+20].

CHAPTER 4

MISUSE SECURITY

In this chapter, we develop new security notions for authenticated encryption schemes with associated data against related-key attacks and analyse generic constructions with respect to these notions. Furthermore, we show that security notions for resetting attacks, a specific case of related-randomness attacks, can be simplified to a single challenge query, which was claimed before but the proof was identified as flawed. The former is based on [FKOS22] while the latter is based on [KS20c].

Contents

4.1 Related-Key Attack Security Notions 106
4.2 Related-Key Attack Security Analysis 109
4.3 Public Key Encryption Vulnerable to Resetting Attacks . . 128
4.4 Security Notions against Resetting Attacks 135
4.5 Summary and Outlook 147

The security of cryptographic primitives crucially relies on randomness. For authenticated encryption, good randomness is needed when generating the secret key, while the actual encryption uses nonces and hence does not rely on good randomness but the much simpler property that nonces are unique. Public key encryption, on the other hand, relies on good randomness during encryption which otherwise allows for re-encrypting attacks. When the randomness is predictable, an adversary can simply obtain the same secret key in case of AEAD schemes or the same random coins for PKE schemes. This allows for trivial attacks and makes security elusive.

A different, yet similar question—and the scope of this chapter—is how security is affected when the randomness is good, but somehow related. This defines the research areas of *related-key attacks* (RKA) and *related-randomness attacks* (RRA).

Related-key attacks are a significant threat to security as demonstrated by a long line of research [BK03, BC10, ABPP18, BF15, AHI11, BR14, Xag13, LLJ15, HLL16, BKN09, KHK11, DKK07, Iso11, Bih94, Knu93, BDK05, KK03, BK09]. The first work that provided a formal model for related-key attacks is the seminal work of Bellare and Kohno [BK03]. In this model, the related-key attacker can specify a related-key-deriving (RKD) function ϕ together with each black-box query to the cryptographic primitive, and observe the input/output behaviour of the primitive under the related key $\phi(K)$. For instance, consider a PRF $\mathcal{F}(K, \cdot)$, that the adversary can query on some input X. As a result of a related-key attack the adversary receives $\mathcal{F}(\phi(K), X)$, where ϕ is the RKD function. Starting with [BK03], several works extend the notion of RKA security to a wide range of cryptographic primitives. This includes pseudorandom functions [BC10, ABPP18], pseudorandom permutations [BF15], encryption schemes [AHI11], and MACs [BR14, Xag13].

Somewhat surprisingly, RKA security has not been considered for the important case of authenticated encryption schemes with associated data. AEAD is a fundamental cryptographic primitive used, e.g., to secure communication in the Internet and is therefore ubiquitously deployed, especially in TLS 1.3 [Res18]. Lately, AEAD has received a lot of attention, for instance through the CAESAR competition [Ber14] and the ongoing NIST standardization process on lightweight cryptography [NIST15]. An important type of AEAD schemes, and simultaneously the focus of the NIST standardization process [NIST15], are so-called nonce-based schemes [Rog02]. These schemes have the advantage that they are deterministic, and hence their security does not rely on good quality randomness during encryption. Instead, they use nonces (e.g., a simple counter) and require that these nonces are never repeated to guarantee security [Rog02].

Related-randomness attacks, introduced by Paterson et al. [PSS14], are similar to related-key attacks in that the adversary can specify a function ϕ (from some set Φ) for each query and receive a ciphertext obtained by encrypting a message using random coins $\phi(r)$ for some initially chosen random coins r. A special case—and simultaneously the inspiration—of RRA are *resetting attacks* (RA) proposed by Yilek [Yil10b] which also define the scope of this work. In case of RA, the set Φ is a singleton containing the identity function, hence any ciphertext that an adversary obtains as part of a resetting attack is generated using the same random coins r.

Security should always hold with respect to adversaries making multiple queries to the challenge oracle. For IND-CPA-like security notions, however, public key encryption schemes are often proven secure against adversaries making only one query to the challenge oracle [KL20, Gol09]. It is folklore that this implies security in the desired case of multiple queries via a standard hybrid argument [MF21]. Yilek [Yil10b] argued that the same holds also for resetting attacks and provides a proof sketch for the claim. Later, however, Paterson et al. [PSS14] pointed out that the proof is flawed as it results in a prohibited query by the reduction. They further argued that they could neither prove Yilek's claim nor give a separation to disprove it and explicitly define their RRA notion in a way that the adversary is allowed to make multiple queries. Thus, prior to this work, it has been unclear whether security against a single challenge query implies security against multiple challenge queries both in the light of resetting attacks and related-randomness attacks.

Contribution

With respect to related-key attacks (RKA), our contribution is as follows. We extend the notion of RKA security to (nonce-based) AEAD schemes. We study the common generic composition paradigms to construct AEAD from encryption schemes and MACs, and explore if RKA security of the underlying primitives carries over to the AEAD scheme. More concretely, let K_e and K_m be the keys of the encryption and MAC, respectively. Assuming that the encryption scheme is secure against the class Φ_e of related-key deriving functions and the MAC is secure against Φ_m, then we ask if the AEAD scheme is secure with respect to related-key derivation functions from the Cartesian product $\Phi_e \times \Phi_m$.[18] In particular, we show that under certain restrictions of $\Phi_e \times \Phi_m$, the schemes N1, N2, and N3 by Namprempre et al. [NRS14], falling into the composition paradigms E&M, EtM, and MtE, respectively, are secure under related-key attacks. By giving concrete attacks against all schemes in case the restrictions are dropped, we show further that these restrictions are necessary. Finally, we show that the RKA security of the encryption scheme $\textsc{Se}[\mathcal{F}, \mathcal{G}]$ and the message authentication code $\textsc{Mac}[\mathcal{H}, \mathcal{F}']$, introduced in Chapter 3, mainly boils down to the RKA security of the underlying functions $\mathcal{F}$ and $\mathcal{F}'$, respectively.

With respect to related-randomness attacks (RRA), our contribution is as follows. We revisit the resetting attack (RA) model proposed by Yilek [Yil10b], which defines a special case of RRA. We prove several schemes insecure with respect to resetting attacks. To this end, we define a class of encryption schemes that we show to be insecure and prove that the concrete schemes lie in this class. As our main contribution, we close the aforementioned gap, by showing that security can be simplified to a single challenge query even against RA, hereby confirming the claim made in [Yil10b] for which the proof was pointed out as flawed in [PSS14]. More precisely, we first investigate the flawed proof in [Yil10b] and give an adversary that distinguishes the different hybrid games in the proof almost perfectly. We then nevertheless prove the claim by giving a different proof approach, which only yields an additional factor of 2 compared to the claimed bound in [Yil10b]. Finally, we define real-or-random security against resetting attacks and prove the equivalence between the existing left-or-right and our new real-or-random security notion, thereby showing that another well-known equivalence from the standard setting also holds in the RA setting.

Related Work

Related-Key Attacks. Based on the initial work by Biham [Bih94] and Knudsen [Knu93], the first formalisation of RKA security has been given by Bellare and Kohno [BK03]. They studied pseudorandom functions as well as pseudorandom permutations and showed an inherit limitation on the set of allowed RKD functions. Bellare and Cash [BC10] proposed RKA-secure pseudorandom functions based on the DDH assumption, which allowed a large class of RKD functions. Abdalla et al. [ABPP18] further increased the allowed class of RKD functions. Several other works study the RKA security for various primitives, e.g., pseudorandom permutations from Feistel networks [BF15], encryption schemes [AHI11], and MACs [BR14, Xag13]. Harris [Har11], and later Albrecht et al. [AFPW11], showed

[18] A similar question using the Cartesian product of the related-key deriving functions from the underlying primitives is answered in [BF15] for Feistel constructions.

inherent limitations of the Bellare-Kohno formalism by giving a generic attack against encryption schemes if the set of related-key deriving functions can depend on the primitive in question. The practical relevance of the alternative model by Harris has been questioned by Vaudenay [Vau16].

Closer to our setting is the work by Lu et al. [LLJ15], who also study RKA security for authenticated encryption schemes. However, instead of nonce-based authenticated encryption schemes, they analyse probabilistic authenticated encryption schemes and only for the specific case of affine functions. Moreover, Han et al. [HLL16] found their proof to be flawed, invalidating the results. To the best of our knowledge, these are the only works that consider RKA security for authenticated encryption schemes.

The practical relevance of RKA security has been shown by a number of works [BKN09, KHK11, DKK07, Iso11], which present attacks against concrete primitives.

Related-Randomness Attacks. Garfinkel and Rosenblum [GR05] pointed out a theoretical threat to the security of a virtual machine, due to the possibility of snapshots. The practical relevance of this threat has later been shown by Ristenpart and Yilek [RY10], who demonstrated attacks on TLS [DA99]. Based on these, Yilek [Yil10b] defined security against resetting attacks, which models the threat pointed out in [GR05]. Later, Paterson et al. [PSS14] and Matsuda and Schuldt [MS18] generalised this to security against related-randomness attacks.

Bellare et al. [BBN+09] gave a public key encryption scheme, which still achieves a meaningful security notion, yet qualitatively worse than IND-CPA, even if the randomness is bad. The same setting is also considered by Bellare and Hoang [BH15] while Kamara and Katz [KK08] study a similar setting for symmetric key encryption schemes. Huang et al. [HLC+18] study nonce-based public key encryption schemes in order to avoid the issue of bad randomness. Closer to our setting is the work by Yang et al. [YDW+12], which studies both resetting attacks and bad randomness, yet for authenticated key exchange.

Roadmap

In Section 4.1 we do define RKA security notions for authenticated encryption as well as for nonce-based symmetric encryption. The RKA security of N-schemes is analysed in Section 4.2, which also analyses the RKA security of the symmetric encryption scheme $\textsc{Se}[\mathcal{F}, \mathcal{G}]$ and message authentication code $\textsc{Mac}[\mathcal{H}, \mathcal{F}']$. We define a class of schemes that are vulnerable to resetting attacks in Section 4.3 and show several schemes that are covered by this class. In Section 4.4 we show that the simplification to a single challenge also holds for resetting attack, thereby restoring the claim in [Yil10b]. Section 4.5 concludes with a summary and outlook for future work.

4.1 Related-Key Attack Security Notions

In this section, we define security for nonce-based encryption schemes and nonce-based AEAD schemes under related-key attacks. RKA security notions for encryption and authenticated encryption schemes have been proposed by Bellare et al. [BCM11] and Lu et

al. [LLJ15], respectively. However, neither notion considers nonce-based primitives; instead both consider the case of probabilistic primitives. Furthermore, both works define indistinguishability in a left-or-right sense, while we follow the stronger IND$ (indistinguishability from random bits) approach put forth by Rogaway [Rog04b]. For this notion, the adversary has to distinguish the encryption of a message from randomly chosen bits. We discuss how the classical property of nonce-respecting adversaries is extended to the RKA setting in Section 4.1.1 and provide two RKA security notions for nonce-based AEAD schemes in Section 4.1.2. Section 4.1.3 provides the RKA security notion for nonce-based encryption schemes.

4.1.1 Nonce Selection

Security notions in the classical setting are often restricted to adversaries which are nonce-respecting. These are adversaries that never repeat a nonce across their encryption queries. Hence, security proven against nonce-respecting adversaries guarantees security as long as the encrypting party never repeats a nonce. Below we argue why this adversarial restriction needs to be updated in the RKA setting.

Consider the following scenario: Alice and Bob communicate using an AEAD scheme across several sessions. In each session, Alice will send several encrypted messages to Bob, each time using a fresh nonce implemented as a counter. Instead of exchanging a fresh secret key for each session, they exchange a key for the first session and between two consecutive sessions, they update the key using some update function $\mathcal{F}$. There is no guarantee that Alice does not reuse a nonce in different sessions. In fact, due to using a simple counter, which might be reset between the sessions, this is likely to happen. This means that an adversary can observe encryptions using the same nonce under related keys, where the relation is given by the update function $\mathcal{F}$.

Another scenario are devices with related keys. As a simple example consider a manufacturer that has some master key K. Rather than generating a fresh key for each device, it derives the key from the master key and some device ID—for instance XORing the two. In this case different users would use different, yet related keys but each user ensures unique nonces only with respect to its own key.

If we declare an RKA adversary to be nonce-respecting if and only if it never repeats a nonce, then a proof of security does not tell us anything for the scenarios described above. Instead, we define an RKA adversary to be RKA-nonce-respecting if it never repeats the pair of nonce and RKD function. An interpretation of this definition is that nonce-respecting is defined with respect to individual keys. If the set of RKD functions is a singleton containing only the identity function, this boils down to the classical definition of nonce-respecting adversaries as there is only one key.

4.1.2 RKA-Security Notions for AEAD Schemes

We extend security for AEAD schemes to the RKA setting. Instead of the approach used in [LLJ15], which defines two separate RKA security notions for confidentiality and authenticity, we follow the unified security notion by Rogaway and Shrimpton [RS06a]. That is, the adversary has access to two oracles Enc and Dec. The goal of the adversary is

to distinguish the real world, in which the oracles implement the encryption and decryption algorithm, from the ideal world, where the first oracle returns random bits while the latter rejects any ciphertext. The adversary wins the game if it can distinguish in which world it is. To make our new RKA security notion achievable, we impose standard restrictions on the adversary. That is, first, the adversary is not allowed to forward the response of an encryption query to the decryption query and, second, the adversary must not repeat a query to its encryption oracle.[19] More precisely, we say that an adversary forwards a query from its encryption oracle, if it queries its decryption oracle on a ciphertext C that it has obtained as a response from its encryption oracle, while the other queried values N, A, ϕ are the same for both queries. We call the resulting notion s-RKA-AE, the "s" indicating strong. The reason for that is that we introduce a weaker notion below.

Definition 4.1.1 (s-RKA-AE Security)**.** *Let $\Sigma = (\mathtt{Enc}, \mathtt{Dec})$ be an AEAD scheme and $\Phi \subset \mathsf{Func}(\mathcal{K}, \mathcal{K})$. Let the game* s-rka-AE *be defined as in Figure 4.1. For an RKA-nonce-respecting and Φ-restricted RKA adversary $\mathcal{A}$, that never repeats/forwards a query to/from* Enc*, we define its* RKA-AE *advantage as*

$$\mathbf{Adv}^{\mathsf{s\text{-}rka\text{-}AE}}_{\Sigma}(\mathcal{A}, \Phi) := \left|2 \Pr[\mathsf{s\text{-}rka\text{-}AE}^{\mathcal{A}} \Rightarrow 1] - 1\right| .$$

Game s-rka-AE	**oracle** $\mathsf{Enc}(N, A, M, \phi)$	**oracle** $\mathsf{Dec}(N, A, C, \phi)$
$b \leftarrow_{\$} \{0,1\}$	**if** $b = 0$	**if** $b = 0$
$K \leftarrow_{\$} \mathcal{K}$	$C \leftarrow_{\$} \{0,1\}^{\gamma}$	$M \leftarrow \perp$
$b' \leftarrow \mathcal{A}^{\mathsf{Enc},\mathsf{Dec}}()$	**else**	**else**
return $(b' = b)$	$C \leftarrow \mathtt{Enc}(\phi(K), N, A, M)$	$M \leftarrow \mathtt{Dec}(\phi(K), N, A, C)$
	return C	**return** M

Figure 4.1: Security game s-rka-AE.

The above definition treats the AEAD scheme to have a single key K. Such schemes, however, are often constructed from smaller building blocks which have individual keys. This encompasses the N constructions [NRS14], on which we focus in the next section, but also all other constructions combining an encryption scheme and a MAC into an AEAD schemes. In this case, the set of RKD functions of the AEAD scheme is the Cartesian product of the set of RKD functions for the individual primitives. More precisely, let E and M be the underlying primitives and Φ_e and Φ_m be the respective sets of RKD functions. Then for the combined primitive AE, the set of RKD functions is $\Phi_{ae} = \Phi_e \times \Phi_m$. Thus, the s-RKA-AE security game above allows the adversary to query the encryption oracle on $(N, \phi_e, \phi_m) \in \mathcal{N} \times \Phi_e \times \Phi_m$ and later querying it on $(N, \phi_e, \phi'_m) \in \mathcal{N} \times \Phi_e \times \Phi_m$, where $\phi_m \neq \phi'_m$. This essentially allows the adversary to bypass the nonce-respecting property of the underlying primitive.

Recall the key-update scenario described above. Allowing the adversary to query the same nonce while the queried RKD functions agree in exactly one part, models a scenario

[19] The latter restriction can also be handled by letting the encryption oracle return the same response as it did when the query was made the first time. For ease of exposition, we simply forbid such queries to avoid additional bookkeeping in the security games.

in which the key update either does not update one of the keys or later updates a key to a previously used key. Similar, for the related devices, where such a query corresponds to the case that only one of the keys is changed or a device ID for one component repeats. We introduce a weaker security notion, in which these queries are forbidden. Security according to this notion then reflects security as long as *the pair of keys* is updated appropriately.

Definition 4.1.2 (RKA-AE Security)**.** *Let* $\Sigma = (\mathtt{Enc}, \mathtt{Dec})$ *be an AEAD scheme and* $\Phi = \Phi_e \times \Phi_m \subset \mathsf{Func}(\mathcal{K}, \mathcal{K})$*. Let the game* rka-AE *be defined as in Figure 4.2. For an RKA-nonce-respecting and* Φ*-restricted RKA adversary* $\mathcal{A}$*, that never repeats/forwards a query to/from* Enc*, we define its* RKA-AE *advantage as:*

$$\mathbf{Adv}^{\mathsf{rka\text{-}AE}}_{\Sigma}(\mathcal{A}, \Phi) \coloneqq \left| 2 \Pr[\mathsf{rka\text{-}AE}^{\mathcal{A}} \Rightarrow 1] - 1 \right| .$$

The weaker security notion RKA-AE bears similarities to split-state non-malleable codes [AAG+16]. Here, the secret is encoded in such a way that it is secure against fault attacks as long as the left and right half of the code are tampered independently. In more detail, the decoding of such tampered codes is independent from the original secret and might be invalid. However, if we consider key-related devices or bad-key updates, non-malleable codes are not helpful any more since they are used for faults and not bad randomised keys. The reason for this is that after each key update we need to take care that the resulting key is still valid. Further, we do not want to update the keys independently but simultaneously such that all keys are fresh after the key update. The requirement to the weaker notion is the opposite of that of non-malleable codes. For key updates, it is a reasonable assumption to say that all underlying keys have to be updated for a new session.

4.1.3 RKA-Security Notions for Symmetric Encryption

The following definition extends the classical IND-CPA security notion for nonce-based encryption schemes to the RKA setting. The adversary has to tell apart the real encryption oracle from an idealised encryption oracle which returns random bits. The main distinction lies in the nonce selection of the adversary as it is allowed to repeat a nonce if the RKD functions are different.

Definition 4.1.3 (IND-CPRKA Security)**.** *Let* $\Sigma = (\mathtt{Enc}, \mathtt{Dec})$ *be an encryption scheme and* $\Phi \subset \mathsf{Func}(\mathcal{K}, \mathcal{K})$*. Let the game* INDCPRKA *be defined as in Figure 4.3. For an RKA-nonce-respecting and* Φ*-restricted RKA adversary* $\mathcal{A}$*, that never repeats a query, we define its* IND-CPRKA *advantage as*

$$\mathbf{Adv}^{\mathsf{INDCPRKA}}_{\Sigma}(\mathcal{A}, \Phi) \coloneqq \left| 2 \Pr[\mathsf{INDCPRKA}^{\mathcal{A}} \Rightarrow 1] - 1 \right| .$$

4.2 Related-Key Attack Security Analysis

In this section we study the security of the nonce-based AEAD schemes N1, N2, and N3 [NRS14], which fall into the generic composition paradigms Encrypt-and-MAC (E&M),

Game rka-AE

$b \leftarrow_\$ \{0,1\}$
$(K_e \parallel K_m) \leftarrow_\$ \mathcal{K}$
$\mathcal{S} \leftarrow \emptyset$
$b' \leftarrow \mathcal{A}^{\mathsf{Enc},\mathsf{Dec}}()$
return $(b' = b)$

oracle $\mathsf{Enc}(N, A, M, \phi_e, \phi_m)$

if $\exists \phi'_e \neq \phi_e$ **s.t.** $(N, \phi'_e, \phi_m) \in \mathcal{S}$
  return $\perp$
if $\exists \phi'_m \neq \phi_m$ **s.t.** $(N, \phi_e, \phi'_m) \in \mathcal{S}$
  return $\perp$
$\mathcal{S} \leftarrow_\cup \{(N, \phi_e, \phi_m)\}$
if $b = 0$
  $C \leftarrow_\$ \{0,1\}^\gamma$
else
  $C \leftarrow \mathtt{Enc}(\phi_e(K_e) \parallel \phi_m(K_m), N, A, M)$
return C

oracle $\mathsf{Dec}(N, A, C, \phi_e, \phi_m)$

if $\exists \phi'_e \neq \phi_e$ **s.t.** $(N, \phi'_e, \phi_m) \in \mathcal{S}$
  return $\perp$
if $\exists \phi'_m \neq \phi_m$ **s.t.** $(N, \phi_e, \phi'_m) \in \mathcal{S}$
  return $\perp$
$\mathcal{S} \leftarrow_\cup \{(N, \phi_e, \phi_m)\}$
if $b = 0$
  $M \leftarrow \perp$
else
  $M \leftarrow \mathtt{Dec}(\phi_e(K_e) \parallel \phi_m(K_m), N, A, M)$
return M

Figure 4.2: Security game rka-AE. The set $\mathcal{S}$ is used to detect forbidden queries, i.e., queries where the triple of nonce and the two RKD functions differ in exactly one of the functions. Both oracles reject such queries by returning $\perp$.

Game INDCPRKA

$b \leftarrow_\$ \{0,1\}$
$K \leftarrow_\$ \mathcal{K}$
$b' \leftarrow \mathcal{A}^{\mathsf{Enc}}()$
return $(b' = b)$

oracle $\mathsf{Enc}(N, M, \phi)$

if $b = 0$
  $C \leftarrow_\$ \{0,1\}^\gamma$
else
  $C \leftarrow \mathtt{Enc}(\phi(K), N, M)$
return C

Figure 4.3: Security game INDCPRKA.

Encrypt-then-MAC (EtM), MAC-then-Encrypt (MtE) [BN00]. We analyse each scheme with respect to the two security notions RKA-AE and s-RKA-AE defined above. The analysis reveals that all schemes achieve RKA-AE security if the underlying primitives are RKA-secure. Regarding the stronger s-RKA-AE security, the situation is more involved. We show that both N1 and N2 are insecure irrespective of the underlying primitives. For N3, we provide a concrete attack exploiting any instantiation using a stream cipher for the underlying encryption scheme.

We further analyse the symmetric encryption scheme $\textsc{Se}[\mathcal{F},\mathcal{G}]$ and message authentication code $\textsc{Mac}[\mathcal{H},\mathcal{F}']$ from Chapter 3. For both constructions, we show that RKA security boils down to the RKA security of the underlying function.

Section 4.2.1 covers the analysis of the N1 construction. The N2 construction is analysed in Section 4.2.2 while we analyse the N3 construction in Section 4.2.3. Section 4.2.4 and Section 4.2.5 analyse the RKA security of $\textsc{Se}[\mathcal{F},\mathcal{G}]$ and $\textsc{Mac}[\mathcal{H},\mathcal{F}']$, respectively.

4.2.1 Related-Key Attack Security of N1

The N1 construction composes a nonce-based encryption scheme and a MAC into an AEAD scheme. It follows the E&M paradigm and is displayed in Figure 2.6. The encryption algorithm is used to encrypt the message as is the MAC to compute a tag for the message. The ciphertext of the AEAD scheme consists of the ciphertext and the tag.

The following theorem shows that the N1 construction achieves RKA-AE security if the underlying primitives are RKA-secure. The overall proof approach is similar to the classical setting but needs some extra treatment when analysing that all queries of the reductions are permitted.

Theorem 4.2.1. *Let $\Sigma = (\mathtt{Enc},\mathtt{Dec})$ be an encryption scheme and $\Gamma = (\mathtt{Tag},\mathtt{Ver})$ be a MAC with RKA function sets Φ_e and Φ_m, respectively. Further, let* N1 *be the AEAD scheme built from Σ and Γ using the N1 construction with RKA function set $\Phi_{ae} = \Phi_e \times \Phi_m$. Then for any RKA-nonce-respecting and Φ_{ae}-restricted RKA adversary $\mathcal{A}$ against* N1*, that never repeats/forwards a query to/from* Enc*, there exists an RKA-nonce-respecting and Φ_e-restricted RKA adversary $\mathcal{A}_{\mathsf{se}}$, a Φ_m-restricted RKA adversary $\mathcal{A}_{\mathsf{mac}}$, and a Φ_m-restricted RKA adversary $\mathcal{A}_{\mathsf{prf}}$ such that*

$$\begin{aligned}\mathbf{Adv}^{\mathsf{rka\text{-}AE}}_{\mathrm{N1}}(\mathcal{A},\Phi_{ae}) \leq\ & \mathbf{Adv}^{\mathsf{INDCPRKA}}_{\Sigma}(\mathcal{A}_{\mathsf{se}},\Phi_e) + \mathbf{Adv}^{\mathsf{SUFCMRKA}}_{\Gamma}(\mathcal{A}_{\mathsf{mac}},\Phi_m) \\ & + \mathbf{Adv}^{\mathsf{rkaPRF}}_{\mathsf{Tag}}(\mathcal{A}_{\mathsf{prf}},\Phi_m)\,.\end{aligned}$$

Proof. We prove the theorem using the hybrid games G_0, G_1, G_2, and G_3, displayed in Figure 4.4 (we drop the set $\mathcal{S}$ to detect forbidden queries for ease of exposition). Game G_0 is rka-AE instantiated with N1 and secret bit b fixed to 1. In G_1, the decryption oracle is modified to reject any ciphertext. In G_2, the tag appended to the ciphertext is chosen at random. Game G_3 equals rka-AE with secret bit b fixed to 0, where the encryption oracle

outputs random ciphertexts and the decryption oracle rejects any ciphertext. It holds that

$$\begin{aligned}\mathbf{Adv}^{\mathsf{rka\text{-}AE}}_{\mathrm{N1}}(\mathcal{A}) &= \Big|\Pr[\mathcal{A}^{\mathsf{rka\text{-}AE}} \Rightarrow 1 \mid b = 1] - \Pr[\mathcal{A}^{\mathsf{rka\text{-}AE}} \Rightarrow 1 \mid b = 0]\Big| \\ &= \Big|\Pr[\mathcal{A}^{\mathsf{G}_0} \Rightarrow 1] - \Pr[\mathcal{A}^{\mathsf{G}_3} \Rightarrow 1]\Big| \\ &\leq \sum_{i=1}^{3} \Big|\Pr[\mathcal{A}^{\mathsf{G}_{i-1}} \Rightarrow 1] - \Pr[\mathcal{A}^{\mathsf{G}_i} \Rightarrow 1]\Big| \,.\end{aligned}$$

To bound the first game hop, we construct an adversary $\mathcal{A}_{\mathsf{mac}}$ playing SUFCMRKA. It picks a key K_e for the encryption scheme at random and runs adversary $\mathcal{A}$ answering queries as follows. Encryption queries of the form $(N, A, M, (\phi_e, \phi_m))$ are processed by computing $C_e \leftarrow \mathtt{Enc}(\phi_e(K_e), N, M)$, querying (N, A, M, ϕ_m) to Tag to obtain T, and sending $C \leftarrow C_e \parallel T$ back to $\mathcal{A}$. For decryption queries $(N, A, C, (\phi_e, \phi_m))$, $\mathcal{A}_{\mathsf{mac}}$ parses C as $C_e \parallel T$, locally computes $M \leftarrow \mathtt{Dec}(\phi_e(K_e), N, C_e)$, and queries Vfy on (N, A, M, T, ϕ_m) to get V. If $V = \top$, $\mathcal{A}_{\mathsf{mac}}$ sends M back to $\mathcal{A}$. If $V = \bot$, $\mathcal{A}_{\mathsf{mac}}$ sends $\bot$ back to $\mathcal{A}$. When $\mathcal{A}$ terminates by outputting a bit b', $\mathcal{A}_{\mathsf{mac}}$ outputs the same bit.

It holds that $\mathcal{A}_{\mathsf{mac}}$ perfectly simulates game G_0 and G_1 based on its own challenge bit from game SUFCMRKA being 1 and 0, respectively. Let $(N, A, C, (\phi_e, \phi_m))$, with $C = C_e \parallel T$, be the decryption query that allows $\mathcal{A}$ to distinguish. For this query $\mathcal{A}_{\mathsf{mac}}$ invokes its oracle Vfy on (N, A, M, T, ϕ_m). It remains to argue that this query is not forbidden, i.e., $\mathcal{A}_{\mathsf{mac}}$ did not query its oracle Tag on (N, A, M, ϕ_m), where $M \leftarrow \mathtt{Dec}(\phi_e(K_e), N, C_e)$, resulting in T. Assume for sake of contradiction, that $\mathcal{A}_{\mathsf{mac}}$ did query (N, A, M, ϕ_m) to Tag. By construction, this means that $\mathcal{A}$ has made a query $(N, A, M, (\phi'_e, \phi_m))$ to Enc, such that $C_e = \mathtt{Enc}(\phi'_e(K_e, N, M))$, for which the response was $C_e \parallel T$. Note that both cases $\phi'_e = \phi_e$ and $\phi'_e \neq \phi_e$ are forbidden queries by $\mathcal{A}$. Thus, we conclude with

$$\begin{aligned}\Big|\Pr[\mathcal{A}^{\mathsf{G}_0} \Rightarrow 1] - \Pr[\mathcal{A}^{\mathsf{G}_1} \Rightarrow 1]\Big| &= \Big|\Pr[\mathcal{A}^{\mathsf{SUFCMRKA}}_{\mathsf{mac}} \Rightarrow 1 \mid b = 1] - \Pr[\mathcal{A}^{\mathsf{SUFCMRKA}}_{\mathsf{mac}} \Rightarrow 1 \mid b = 0]\Big| \\ &= \mathbf{Adv}^{\mathsf{SUFCMRKA}}_{\Gamma}(\mathcal{A}_{\mathsf{mac}}, \Phi_m) \,.\end{aligned}$$

For the remaining two game hops, the decryption oracle always responds with $\bot$ which is why we omit it in the description. To bound the advantage between games G_1 and G_2 we construct an adversary $\mathcal{A}_{\mathsf{prf}}$ against Tag. It picks a key K_e for the encryption scheme and runs $\mathcal{A}$. For each query $(N, A, M, (\phi_e, \phi_m))$ that $\mathcal{A}$ makes to Enc, $\mathcal{A}_{\mathsf{prf}}$ locally computes $C_e \leftarrow \mathtt{Enc}(\phi_e(K_e), N, M)$, and queries (N, A, M, ϕ_m) to F to get T. Then it sends $C \leftarrow C_e \parallel T$ back to $\mathcal{A}$. Finally, $\mathcal{A}_{\mathsf{prf}}$ outputs whatever $\mathcal{A}$ outputs.

It holds that $\mathcal{A}_{\mathsf{prf}}$ perfectly simulates G_1 for $\mathcal{A}$ if its own challenge oracle is initialised with $b = 1$. Likewise, $\mathcal{A}_{\mathsf{prf}}$ perfectly simulates G_2 if its own challenge oracle is initialised with $b = 0$. Also, $\mathcal{A}_{\mathsf{prf}}$ never repeats a query. Every query (N, A, M, ϕ_m) by $\mathcal{A}_{\mathsf{prf}}$ stems from a query $(N, A, M, (\phi_e, \phi_m))$ by $\mathcal{A}$. This means that $\mathcal{A}_{\mathsf{prf}}$ repeats a query either if $\mathcal{A}$ repeats a query or $\mathcal{A}$ makes two queries differing only in the RKD function for the encryption scheme, which are both forbidden. This yields

$$\begin{aligned}\Big|\Pr[\mathcal{A}^{\mathsf{G}_1} \Rightarrow 1] - \Pr[\mathcal{A}^{\mathsf{G}_2} \Rightarrow 1]\Big| &= \Big|\Pr[\mathcal{A}^{\mathsf{rkaPRF}}_{\mathsf{prf}} \Rightarrow 1 \mid b = 1] - \Pr[\mathcal{A}^{\mathsf{rkaPRF}}_{\mathsf{prf}} \Rightarrow 1 \mid b = 0]\Big| \\ &= \mathbf{Adv}^{\mathsf{rkaPRF}}_{\mathsf{Tag}}(\mathcal{A}_{\mathsf{prf}}, \Phi_m) \,.\end{aligned}$$

To bound the last game hop, we construct the following adversary $\mathcal{A}_{\mathsf{se}}$ against Σ playing INDCPRKA. It runs $\mathcal{A}$ and answers its queries $(N, A, M, (\phi_e, \phi_m))$ to Enc as follows. It picks T at random of appropriate length, queries its own oracle Enc on (N, M, ϕ_e) to obtain C_e and it sends $C \leftarrow C_e \parallel T$ to $\mathcal{A}$. At the end, $\mathcal{A}_{\mathsf{se}}$ outputs the output of $\mathcal{A}$.

If its own challenge bit b equals 1, $\mathcal{A}_{\mathsf{se}}$ simulates game G_2 for $\mathcal{A}$, if the bit b equals 0, it simulates game G_3 for $\mathcal{A}$. Thus, we get

$$\begin{aligned}\left|\Pr[\mathcal{A}^{\mathsf{G}_2} \Rightarrow 1] - \Pr[\mathcal{A}^{\mathsf{G}_3} \Rightarrow 1]\right| &= \left|\Pr[\mathcal{A}_{\mathsf{se}}^{\mathsf{INDCPRKA}} \Rightarrow 1 \mid b = 1] - \Pr[\mathcal{A}_{\mathsf{se}}^{\mathsf{INDCPRKA}} \Rightarrow 1 \mid b = 0]\right| \\ &= \mathbf{Adv}_{\Sigma}^{\mathsf{INDCPRKA}}(\mathcal{A}_{\mathsf{se}}, \Phi_e)\,.\end{aligned}$$

Collecting the bounds from the individual game hops proves the claim. □

Game G_i

$(K_e, K_m) \leftarrow_{\$} \mathcal{K}$
$b' \leftarrow \mathcal{A}^{\mathsf{Enc},\mathsf{Dec}}()$

oracle $\mathsf{Dec}(N, A, C, (\phi_e, \phi_m))$ in G_0

$C_e \parallel T \leftarrow C$
$M \leftarrow \mathtt{Dec}(\phi_e(K_e), N, C_e)$
if $\mathtt{Ver}(\phi_m(K_m), N, A, M, T) = \top$
  return M
return $\perp$

oracle $\mathsf{Dec}(N, A, C, (\phi_e, \phi_m))$ in G_1, G_2, G_3

return $\perp$

oracle $\mathsf{Enc}(N, A, M, (\phi_e, \phi_m))$ in G_0, G_1

$C_e \leftarrow \mathtt{Enc}(\phi_e(K_e), N, M)$
$T \leftarrow \mathtt{Tag}(\phi_m(K_m), N, A, M)$
$C \leftarrow C_e \parallel T$
return C

oracle $\mathsf{Enc}(N, A, M, (\phi_e, \phi_m))$ in G_2

$C_e \leftarrow \mathtt{Enc}(\phi_e(K_e), N, M)$
$T \leftarrow_{\$} \{0,1\}^{\tau}$
$C \leftarrow C_e \parallel T$
return C

oracle $\mathsf{Enc}(N, A, M, (\phi_e, \phi_m))$ in G_3

$C_e \leftarrow_{\$} \{0,1\}^{\gamma}$
$T \leftarrow_{\$} \{0,1\}^{\tau}$
$C \leftarrow C_e \parallel T$
return C

Figure 4.4: Games G_i used to prove Theorem 4.2.1 (RKA-AE security of N1).

The following theorem shows that the N1 construction does not achieve the stronger s-RKA-AE security. The reason is that a ciphertext is the concatenation of a ciphertext from the underlying encryption scheme and tag from the underlying MAC. By making two queries which solely differ in one of the RKD functions, the adversary can easily distinguish between the real and the ideal case.

Theorem 4.2.2. *Let $\Sigma = (\mathtt{Enc}, \mathtt{Dec})$ be an encryption scheme and $\Gamma = (\mathtt{Tag}, \mathtt{Ver})$ be a MAC with RKA function sets Φ_e and Φ_m, respectively. Further, let* N1 *be the AEAD scheme built from Σ and Γ using the N1 construction with RKA function set $\Phi_{ae} =$*

$\Phi_e \times \Phi_m$. *Then* N1 *is not* s-RKA-AE*-secure. There exists an RKA-nonce-respecting and* Φ_{ae}*-restricted RKA adversary* $\mathcal{A}$ *such that*

$$\mathbf{Adv}^{\mathsf{s\text{-}rka\text{-}AE}}_{\mathrm{N1}}(\mathcal{A}) = 1 - 2^{-y}\,,$$

where $y = \max(\gamma, \tau)$.

Proof. We construct the following two adversaries $\mathcal{A}_{\mathsf{e}}$ and $\mathcal{A}_{\mathsf{m}}$. Adversary $\mathcal{A}_{\mathsf{e}}$ picks a message M, a nonce N, and associated data A at random from the respective sets. Furthermore, it picks $\phi_e \in \Phi_e$ and $\phi_m, \phi'_m \in \Phi_m$ such that $\phi_m \neq \phi'_m$. Then the adversary makes two queries to its encryption oracle Enc: (1) it queries $(N, A, M, (\phi_e, \phi_m))$ and (2) it queries $(N, A, M, (\phi_e, \phi'_m))$. These queries result in two ciphertexts $C_1 = C_{e,1} \parallel T_1$ and $C_2 = C_{e,2} \parallel T_2$. The adversary outputs $b' = 1$ if $C_{e,1} = C_{e,2}$, otherwise, it outputs $b' = 0$.

Note, first, that both queries are valid as they differ in the RKD function of the underlying MAC. If the bit b of game $\mathsf{s\text{-}rka\text{-}AE}$ equals 0, the adversary will receive two randomly chosen bit strings as the ciphertexts C_1 and C_2. Hence, we get

$$\Pr[\mathcal{A}_{\mathsf{e}}^{\mathsf{s\text{-}rka\text{-}AE}} \Rightarrow 1 \,|\, b = 0] = 2^{-\gamma}\,.$$

If the bit b is 1, the adversary will receive $C_1 = C_{e,1} \parallel T_1$, where $C_{e,1} = \mathtt{Enc}(\phi_e(K_e), N, M)$, and $C_2 = C_{e,2} \parallel T_2$, where $C_{e,2} = \mathtt{Enc}(\phi_e(K_e), N, M)$. Since $C_{e,1}$ equals $C_{e,2}$, we get

$$\Pr[\mathcal{A}_{\mathsf{e}}^{\mathsf{s\text{-}rka\text{-}AE}} \Rightarrow 1 \,|\, b = 1] = 1\,.$$

Putting the above together we obtain

$$\begin{aligned}\mathbf{Adv}^{\mathsf{s\text{-}rka\text{-}AE}}_{\mathrm{N1}}(\mathcal{A}_{\mathsf{e}}) &= \left|\Pr[\mathcal{A}_{\mathsf{e}}^{\mathsf{s\text{-}rka\text{-}AE}} \Rightarrow 1 \,|\, b = 1] - \Pr[\mathcal{A}_{\mathsf{e}}^{\mathsf{s\text{-}rka\text{-}AE}} \Rightarrow 1 \,|\, b = 0]\right| \\ &= 1 - 2^{-\gamma}\,.\end{aligned}$$

Adversary $\mathcal{A}_{\mathsf{m}}$ picks a message M, a nonce N, and associated data A at random from the respective sets. Furthermore, it picks $\phi_m \in \Phi_m$ and $\phi_e, \phi'_e \in \Phi_e$ such that $\phi_e \neq \phi'_e$. Then the adversary makes two queries to its encryption oracle Enc: (1) it queries $(N, A, M, (\phi_e, \phi_m))$ and (2) it queries $(N, A, M, (\phi'_e, \phi_m))$. These queries result in two ciphertexts $C_1 = C_{e,1} \parallel T_1$ and $C_2 = C_{e,2} \parallel T_2$. If $T_1 = T_2$, the adversary outputs $b' = 1$, otherwise, it outputs $b' = 0$.

Again, note that both queries are valid as they differ in the RKD function of the underlying encryption scheme. If the bit b of game $\mathsf{s\text{-}rka\text{-}AE}$ equals 0, the adversary will receive two randomly chosen bit strings for C_1 and C_2. This yields

$$\Pr[\mathcal{A}_{\mathsf{m}}^{\mathsf{s\text{-}rka\text{-}AE}} \Rightarrow 1 \,|\, b = 0] = 2^{-\tau}\,.$$

If the bit b is 1, $\mathcal{A}_{\mathsf{m}}$ will receive $C_1 = C_{e,1} \parallel T_1$, where $T_1 = \mathtt{Tag}(\phi_m(K_m), N, A, M)$, and $C_2 = C_{e,2} \parallel T_2$, where $T_2 = \mathtt{Tag}(\phi_m(K_m), N, A, M)$. Since T_1 equals T_2 we get

$$\Pr[\mathcal{A}_{\mathsf{m}}^{\mathsf{s\text{-}rka\text{-}AE}} \Rightarrow 1 \,|\, b = 1] = 1\,.$$

Collecting the above yields

$$\begin{aligned}\mathbf{Adv}^{\mathsf{s\text{-}rka\text{-}AE}}_{\mathrm{N1}}(\mathcal{A}_{\mathsf{m}}) &= \Big|\Pr[\mathcal{A}^{\mathsf{s\text{-}rka\text{-}AE}}_{\mathsf{m}} \Rightarrow 1 \,|\, b = 1] - \Pr[\mathcal{A}^{\mathsf{s\text{-}rka\text{-}AE}}_{\mathsf{m}} \Rightarrow 1 \,|\, b = 0]\Big| \\ &= 1 - 2^{-\tau}\,.\end{aligned}$$

Letting $\mathcal{A}$ be either be $\mathcal{A}_{\mathsf{e}}$, if $\gamma > \tau$, or $\mathcal{A}_{\mathsf{m}}$, if $\gamma < \tau$ proves the claim. Note that τ is fixed while γ is something specified by the adversary via the length of the message it queries. In general, we can assume that $\gamma > \tau$. □

4.2.2 Related-Key Attack Security of N2

The N2 construction composes a nonce-based encryption scheme and a MAC into an AEAD scheme. It follows the EtM paradigm and is displayed in Figure 2.7. The scheme first encrypts the message using the encryption scheme. Subsequently, the MAC is used to compute a tag for the ciphertext. The ciphertext of the AEAD scheme consists of both the ciphertext and the tag.

The theorem below shows that the N2 construction achieves RKA-AE security if the underlying primitives are sound. The overall proof follows the classical one, except for a more complex analysis regarding the permitted queries.

Theorem 4.2.3. *Let $\Sigma = (\mathtt{Enc}, \mathtt{Dec})$ be an encryption scheme and $\Gamma = (\mathtt{Tag}, \mathtt{Ver})$ be a MAC with RKA function sets Φ_e and Φ_m, respectively. Further, let* N2 *be the AEAD scheme built from Σ and Γ using the N2 construction with RKA function set $\Phi_{ae} = \Phi_e \times \Phi_m$. Then for any RKA-nonce-respecting and Φ_{ae}-restricted RKA adversary $\mathcal{A}$ against* N2*, that never repeats/forwards a query to/from* Enc*, there exists an RKA-nonce-respecting and Φ_e-restricted RKA adversary $\mathcal{A}_{\mathsf{se}}$, a Φ_m-restricted RKA adversary $\mathcal{A}_{\mathsf{mac}}$, and a Φ_m-restricted RKA adversary $\mathcal{A}_{\mathsf{prf}}$ such that*

$$\begin{aligned}\mathbf{Adv}^{\mathsf{rka\text{-}AE}}_{\mathrm{N2}}(\mathcal{A}, \Phi_{ae}) \leq\ & \mathbf{Adv}^{\mathsf{INDCPRKA}}_{\Sigma}(\mathcal{A}_{\mathsf{se}}, \Phi_e) + \mathbf{Adv}^{\mathsf{SUFCMRKA}}_{\Gamma}(\mathcal{A}_{\mathsf{mac}}, \Phi_m) \\ & + \mathbf{Adv}^{\mathsf{rkaPRF}}_{\mathsf{Tag}}(\mathcal{A}_{\mathsf{prf}}, \Phi_m)\,.\end{aligned}$$

Proof. We prove the theorem using the hybrid games G_0, G_1, G_2, and G_3 displayed in Figure 4.5. For ease of exposition, the games do not contain the set $\mathcal{S}$ to detect forbidden queries. Instead, we assume that the adversary does not make such queries, in which case the reduction would simply return $\perp$. Game G_0 is rka-AE instantiated with N2 and secret bit b fixed to 1. In G_1, the decryption oracle is modified to reject any ciphertext. In G_2, the tag appended to the ciphertext is chosen at random. Game G_3 equals rka-AE with secret bit b fixed to 0, where the encryption oracle outputs random ciphertexts and the decryption oracle rejects any ciphertext. We have

$$\begin{aligned}\mathbf{Adv}^{\mathsf{rka\text{-}AE}}_{\mathrm{N2}}(\mathcal{A}) &= \Big|\Pr[\mathcal{A}^{\mathsf{rka\text{-}AE}} \Rightarrow 1 \,|\, b = 1] - \Pr[\mathcal{A}^{\mathsf{rka\text{-}AE}} \Rightarrow 1 \,|\, b = 0]\Big| \\ &= \Big|\Pr[\mathcal{A}^{\mathsf{G}_0} \Rightarrow 1] - \Pr[\mathcal{A}^{\mathsf{G}_3} \Rightarrow 1]\Big| \\ &\leq \sum_{i=1}^{3} \Big|\Pr[\mathcal{A}^{\mathsf{G}_{i-1}} \Rightarrow 1] - \Pr[\mathcal{A}^{\mathsf{G}_i} \Rightarrow 1]\Big|\,.\end{aligned}$$

To bound the first game hop, we construct an adversary $\mathcal{A}_{\mathsf{mac}}$ playing SUFCMRKA. It picks a key K_e for the encryption scheme at random and runs adversary $\mathcal{A}$ answering queries as follows. Encryption queries of the form $(N, A, M, (\phi_e, \phi_m))$ are processed by computing $C_e \leftarrow \mathtt{Enc}(\phi_e(K_e), N, M)$, querying (N, A, C_e, ϕ_m) to Tag to obtain T, and sending $C \leftarrow C_e \parallel T$ back to $\mathcal{A}$. For decryption queries $(N, A, C, (\phi_e, \phi_m))$, $\mathcal{A}_{\mathsf{mac}}$ parses C as $C_e \parallel T$ queries Vfy on (N, A, C_e, T, ϕ_m) to get V. If $V = \top$, $\mathcal{A}_{\mathsf{mac}}$ computes $M \leftarrow \mathtt{Dec}(\phi_e(K_e), N, C_e)$ and sends M back to $\mathcal{A}$. If $V = \bot$, $\mathcal{A}_{\mathsf{mac}}$ sends $\bot$ back to $\mathcal{A}$. When $\mathcal{A}$ terminates by outputting a bit b', $\mathcal{A}_{\mathsf{mac}}$ outputs the same bit.

It holds that $\mathcal{A}_{\mathsf{mac}}$ perfectly simulates game G_0 and G_1 based on its own challenge bit from game SUFCMRKA being 1 and 0, respectively. Let $(N, A, C, (\phi_e, \phi_m))$, with $C = C_e \parallel T$, be the decryption query that allows $\mathcal{A}$ to distinguish. For this query $\mathcal{A}_{\mathsf{mac}}$ invokes its oracle Vfy on (N, A, C_e, T, ϕ_m). It remains to argue that this query is not forbidden, i.e., $\mathcal{A}_{\mathsf{mac}}$ did not query its oracle Tag on (N, A, C_e, ϕ_m) resulting in T. Assume for sake of contradiction, that $\mathcal{A}_{\mathsf{mac}}$ did query (N, A, C_e, ϕ_m) to Tag. By construction, this means that $\mathcal{A}$ has made a query $(N, A, M, (\phi'_e, \phi_m))$ to Enc, such that $C_e = \mathtt{Enc}(\phi'_e(K_e, N, M))$ for which the response was $C_e \parallel T$. Note that both cases $\phi'_e = \phi_e$ and $\phi'_e \neq \phi_e$ are forbidden queries by $\mathcal{A}$. Thus we conclude with

$$\begin{aligned}\Big|\Pr[\mathcal{A}^{\mathsf{G}_0} \Rightarrow 1] - \Pr[\mathcal{A}^{\mathsf{G}_1} \Rightarrow 1]\Big| &= \Big|\Pr[\mathcal{A}_{\mathsf{mac}}^{\mathsf{SUFCMRKA}} \Rightarrow 1 \,|\, b = 1] - \Pr[\mathcal{A}_{\mathsf{mac}}^{\mathsf{SUFCMRKA}} \Rightarrow 1 \,|\, b = 0]\Big| \\ &= \mathbf{Adv}_{\Gamma}^{\mathsf{SUFCMRKA}}(\mathcal{A}_{\mathsf{mac}}, \Phi_m)\,.\end{aligned}$$

For the remaining two game hops, we omit the decryption oracle since any reduction merely needs to respond with $\bot$. To bound the advantage between game G_1 and G_2 we construct an adversary $\mathcal{A}_{\mathsf{prf}}$ against Tag. It picks a key K_e for the encryption scheme and runs $\mathcal{A}$. For each query $(N, A, M, (\phi_e, \phi_m))$ that $\mathcal{A}$ makes to Enc, $\mathcal{A}_{\mathsf{prf}}$ locally computes $C_e \leftarrow \mathtt{Enc}(\phi_e(K_e), N, M)$, and queries (N, A, C_e, ϕ_m) to F to get T. Then it sends $C \leftarrow C_e \parallel T$ back to $\mathcal{A}$. Finally, $\mathcal{A}_{\mathsf{prf}}$ outputs whatever $\mathcal{A}$ outputs.

It holds that $\mathcal{A}_{\mathsf{prf}}$ perfectly simulates G_1 for $\mathcal{A}$ if its own challenge oracle is initialised with $b = 1$. Likewise, $\mathcal{A}_{\mathsf{prf}}$ perfectly simulates G_2 if its own challenge oracle is initialised with $b = 0$. This yields

$$\begin{aligned}\Big|\Pr[\mathcal{A}^{\mathsf{G}_1} \Rightarrow 1] - \Pr[\mathcal{A}^{\mathsf{G}_2} \Rightarrow 1]\Big| &= \Big|\Pr[\mathcal{A}_{\mathsf{prf}}^{\mathsf{rkaPRF}} \Rightarrow 1 \,|\, b = 1] - \Pr[\mathcal{A}_{\mathsf{prf}}^{\mathsf{rkaPRF}} \Rightarrow 1 \,|\, b = 0]\Big| \\ &= \mathbf{Adv}_{\mathsf{Tag}}^{\mathsf{rkaPRF}}(\mathcal{A}_{\mathsf{prf}}, \Phi_m)\,.\end{aligned}$$

To bound the last game hop, we construct the following adversary $\mathcal{A}_{\mathsf{se}}$ against Σ playing INDCPRKA. It runs $\mathcal{A}$ and answers its queries $(N, A, M, (\phi_e, \phi_m))$ to Enc as follows. It picks T at random of appropriate length, queries its own oracle Enc on (N, M, ϕ_e) to obtain C_e and sends $C \leftarrow C_e \parallel T$ to $\mathcal{A}$. At the end, $\mathcal{A}_{\mathsf{se}}$ outputs whatever $\mathcal{A}$ outputs.

If its own challenge bit b equals 1, $\mathcal{A}_{\mathsf{se}}$ simulates game G_2 for $\mathcal{A}$, if the bit b equals 0, it simulates game G_3 for $\mathcal{A}$. Thus we get

$$\begin{aligned}\Big|\Pr[\mathcal{A}^{\mathsf{G}_2} \Rightarrow 1] - \Pr[\mathcal{A}^{\mathsf{G}_3} \Rightarrow 1]\Big| &= \Big|\Pr[\mathcal{A}_{\mathsf{se}}^{\mathsf{INDCPRKA}} \Rightarrow 1 \,|\, b = 1] - \Pr[\mathcal{A}_{\mathsf{se}}^{\mathsf{INDCPRKA}} \Rightarrow 1 \,|\, b = 0]\Big| \\ &= \mathbf{Adv}_{\Sigma}^{\mathsf{INDCPRKA}}(\mathcal{A}_{\mathsf{se}}, \Phi_e)\,.\end{aligned}$$

Collecting the bounds from the individual game hops proves the claim. □

Game G_i

$(K_e, K_m) \leftarrow\!\!\$\ \mathcal{K}$
$b' \leftarrow \mathcal{A}^{\mathsf{Enc},\mathsf{Dec}}()$

oracle $\mathsf{Dec}(N, A, C, (\phi_e, \phi_m))$ in G_0

$C_e \parallel T \leftarrow C$
if $\mathtt{Ver}(\phi_m(K_m), N, A, C_e, T) = \top$
  $M \leftarrow \mathtt{Dec}(\phi_e(K_e), N, C_e)$
  return M
return $\perp$

oracle $\mathsf{Dec}(N, A, C, (\phi_e, \phi_m))$ in G_1, G_2, G_3

return $\perp$

oracle $\mathsf{Enc}(N, A, M, (\phi_e, \phi_m))$ in G_0, G_1

$C_e \leftarrow \mathtt{Enc}(\phi_e(K_e), N, M)$
$T \leftarrow \mathtt{Tag}(\phi_m(K_m), N, A, C_e)$
$C \leftarrow C_e \parallel T$
return C

oracle $\mathsf{Enc}(N, A, M, (\phi_e, \phi_m))$ in G_2

$C_e \leftarrow \mathtt{Enc}(\phi_e(K_e), N, M)$
$T \leftarrow\!\!\$\ \{0,1\}^\tau$
$C \leftarrow C_e \parallel T$
return C

oracle $\mathsf{Enc}(N, A, M, (\phi_e, \phi_m))$ in G_3

$C_e \leftarrow\!\!\$\ \{0,1\}^\gamma$
$T \leftarrow\!\!\$\ \{0,1\}^\tau$
$C \leftarrow C_e \parallel T$
return C

Figure 4.5: Games G_i used to prove Theorem 4.2.3 (RKA-AE security of N2).

Below we show that the N2 construction does not achieve s-RKA-AE security. It exhibits the same structure as the N1 construction, that is, a concatenation of a ciphertext and a tag from the underlying primitives. The difference is that the tag is computed on the ciphertext rather than the message. Consequently, one of the attacks against the N1 construction, the one exploiting the encryption scheme, also applies to the N2 construction, while the other, against the MAC, does not apply. Nevertheless, this shows that the N2 construction is insecure.

Theorem 4.2.4. *Let $\Sigma = (\mathtt{Enc}, \mathtt{Dec})$ be an encryption scheme and $\Gamma = (\mathtt{Tag}, \mathtt{Ver})$ be a MAC with RKA function sets Φ_e and Φ_m, respectively. Further, let* N2 *be the AEAD scheme built from Σ and Γ using the N2 construction with RKA function set $\Phi_{ae} = \Phi_e \times \Phi_m$. Then,* N2 *is not* s-RKA-AE*-secure. There exists an RKA-nonce-respecting and Φ_{ae}-restricted RKA adversary $\mathcal{A}$ such that*

$$\mathbf{Adv}^{\mathsf{s\text{-}rka\text{-}AE}}_{\mathrm{N2}}(\mathcal{A}) = 1 - 2^{-\gamma}\,.$$

Proof. We construct the following adversary $\mathcal{A}$. It picks a message M, a nonce N, and associated data A at random from the respective sets. Furthermore, it picks $\phi_e \in \Phi_e$ and $\phi_m, \phi'_m \in \Phi_m$ such that $\phi_m \neq \phi'_m$. Then the adversary makes two queries to its encryption oracle Enc: (1) it queries $(N, A, M, (\phi_e, \phi_m))$ and (2) it queries $(N, A, M, (\phi_e, \phi'_m))$. These queries result in two ciphertexts $C_1 = C_{e,1} \parallel T_1$ and $C_2 = C_{e,2} \parallel T_2$. If $C_{e,1} = C_{e,2}$, the adversary outputs $b' = 1$, otherwise, it outputs $b' = 0$.

Note first that both queries are valid as they differ in the RKD function of the underlying MAC. If the bit b of game $\mathsf{s\text{-}rka\text{-}AE}$ equals 1, the adversary will receive two randomly chosen bit strings as the ciphertexts C_1 and C_2. It holds that

$$\Pr[\mathcal{A}^{\mathsf{s\text{-}rka\text{-}AE}} \Rightarrow 1 \,|\, b = 0] = 2^{-\gamma}\,.$$

If the bit b is 0, $\mathcal{A}$ will receive $C_1 = C_{e,1} \parallel T_1$, where $C_{e,1} = \mathtt{Enc}(\phi_e(K_e), N, M)$, and $C_2 = C_{e,2} \parallel T_2$, where $C_{e,2} = \mathtt{Enc}(\phi_e(K_e), N, M)$. Since $C_{e,1}$ equals $C_{e,2}$ we obtain

$$\Pr[\mathcal{A}^{\mathsf{s\text{-}rka\text{-}AE}} \Rightarrow 1 \,|\, b = 1] = 1\,.$$

Collecting the above yields

$$\begin{aligned}\mathbf{Adv}^{\mathsf{s\text{-}rka\text{-}AE}}_{\mathrm{N1}}(\mathcal{A}) &= \Big|\Pr[\mathcal{A}^{\mathsf{s\text{-}rka\text{-}AE}} \Rightarrow 1 \,|\, b = 1] - \Pr[\mathcal{A}^{\mathsf{s\text{-}rka\text{-}AE}} \Rightarrow 1 \,|\, b = 0]\Big| \\ &= 1 - 2^{-\gamma}\,.\end{aligned}$$

This concludes the proof. □

4.2.3 Related-Key Attack Security of N3

The N3 construction composes a nonce-based encryption scheme and a MAC into an AEAD scheme. It follows the MtE paradigm and is displayed in Figure 2.8. The message is first used as an input to the MAC and then both the message and the tag are encrypted. In contrast to the other compositions, the ciphertext of the AEAD scheme consists only of the ciphertext from the underlying encryption scheme.

In the theorem below, we show that the N3 construction is RKA-AE secure if both of the underlying primitives are secure. The overall proof follows the classical setting, except for the analysis that all queries by the reductions are indeed valid queries.

Theorem 4.2.5. *Let $\Sigma = (\mathtt{Enc}, \mathtt{Dec})$ be an encryption scheme and $\Gamma = (\mathtt{Tag}, \mathtt{Ver})$ be a MAC with RKA function sets Φ_e and Φ_m, respectively. Further, let* N3 *be the AEAD scheme built from Σ and Γ using the N3 construction with RKA function set $\Phi_{ae} = \Phi_e \times \Phi_m$. Then, for any RKA-nonce-respecting and Φ_{ae}-restricted RKA adversary $\mathcal{A}$ against* N3*, that never repeats/forwards a query to/from* Enc*, there exists an RKA-nonce-respecting and Φ_e-restricted RKA adversary $\mathcal{A}_{\mathsf{se}}$, a Φ_m-restricted RKA adversary $\mathcal{A}_{\mathsf{mac}}$, and a Φ_m-restricted RKA adversary $\mathcal{A}_{\mathsf{prf}}$ such that*

$$\begin{aligned}\mathbf{Adv}^{\mathsf{rka\text{-}AE}}_{\mathrm{N3}}(\mathcal{A}, \Phi_{ae}) \leq\ & \mathbf{Adv}^{\mathsf{INDCPRKA}}_{\Sigma}(\mathcal{A}_{\mathsf{se}}, \Phi_e) + \mathbf{Adv}^{\mathsf{SUFCMRKA}}_{\Gamma}(\mathcal{A}_{\mathsf{mac}}, \Phi_m) \\ & + \mathbf{Adv}^{\mathsf{rkaPRF}}_{\mathsf{Tag}}(\mathcal{A}_{\mathsf{prf}}, \Phi_m)\,.\end{aligned}$$

Proof. We prove the theorem using the hybrid games G_0, G_1, G_2, and G_3 displayed in Figure 4.6. For sake of simplicity, the games do not contain the set $\mathcal{S}$ to detect invalid queries. Instead, we assume that the adversary does not make such queries, which the reduction can simply answer with $\perp$. Game G_0 is $\mathsf{rka\text{-}AE}$ instantiated with N3 and secret bit b fixed to 1. In G_1, the decryption oracle is modified to reject any ciphertext. In G_2, encryption oracle computes a random tag which is then encrypted along with the message.

Game G_3 equals rka-AE with secret bit b fixed to 0, where the encryption oracle outputs random ciphertexts and the decryption oracle rejects any ciphertext. We have

$$\begin{aligned}\mathbf{Adv}^{\mathsf{rka\text{-}AE}}_{\mathrm{N3}}(\mathcal{A}) &= \Big|\Pr[\mathcal{A}^{\mathsf{rka\text{-}AE}} \Rightarrow 1 \,|\, b=1] - \Pr[\mathcal{A}^{\mathsf{rka\text{-}AE}} \Rightarrow 1 \,|\, b=0]\Big| \\ &= \Big|\Pr[\mathcal{A}^{\mathsf{G}_0} \Rightarrow 1] - \Pr[\mathcal{A}^{\mathsf{G}_3} \Rightarrow 1]\Big| \\ &\leq \sum_{i=1}^{3} \Big|\Pr[\mathcal{A}^{\mathsf{G}_{i-1}} \Rightarrow 1] - \Pr[\mathcal{A}^{\mathsf{G}_i} \Rightarrow 1]\Big| \,.\end{aligned}$$

To bound the term $\Pr[\mathcal{A}^{\mathsf{G}_0} \Rightarrow 1] - \Pr[\mathcal{A}^{\mathsf{G}_1} \Rightarrow 1]$, we construct the following adversary $\mathcal{A}_{\mathsf{mac}}$ against the SUF-CMRKA security of Γ. It chooses a random key K_e for the encryption scheme Σ and then runs $\mathcal{A}$. When $\mathcal{A}$ makes a query $(N, A, M, (\phi_e, \phi_m))$ to Enc, $\mathcal{A}_{\mathsf{mac}}$ proceeds as follows. It queries its oracle Tag on (N, A, M, ϕ_m) to obtain a tag T. Then it locally computes $C \leftarrow \mathtt{Enc}(\phi_e(K_e), N, M \parallel T)$ and sends C back to $\mathcal{A}$. For queries $(N, A, C, (\phi_e, \phi_m))$ to Dec by $\mathcal{A}$, $\mathcal{A}_{\mathsf{mac}}$ locally computes $M \parallel T \leftarrow \mathtt{Dec}(\phi_e(K_e), N, C)$ and queries (N, A, M, T, ϕ_m) to its challenge oracle Vfy. If the response is $\perp$, it forwards it to $\mathcal{A}$, otherwise, it sends M to $\mathcal{A}$. When $\mathcal{A}$ outputs a bit b', $\mathcal{A}_{\mathsf{mac}}$ outputs the same bit.

It remains to argue that $\mathcal{A}_{\mathsf{mac}}$ never makes a forbidden query (forwarding from Tag to Vfy) conditioned on $\mathcal{A}$ making only permitted queries. Assume, for sake of contradiction, that $\mathcal{A}$ makes a valid query $(N, A, C, \phi_e, \phi_m)$ to Dec for which $\mathcal{A}_{\mathsf{mac}}$ makes a forbidden query. By construction, $\mathcal{A}_{\mathsf{mac}}$ computes $M \parallel T \leftarrow \mathtt{Dec}(\phi_e(K_e), N, C)$ and queries Vfy on (N, A, M, T, ϕ_m). This query is forbidden if $\mathcal{A}_{\mathsf{mac}}$ has queried (N, A, M, ϕ_m) to Tag which resulted in T. This happens if $\mathcal{A}$ has made a query $(N, A, M, \phi'_e, \phi_m)$ to Enc. We need to distinguish between the case $\phi'_e = \phi_e$ and $\phi'_e \neq \phi_e$. The former is forbidden as this means that $\mathcal{A}$ forwards a query from Enc to Dec. The latter is forbidden since game rka-AE forbids queries that agree on the nonce and exactly one of the RKD functions while disagreeing on the other RKD function. Hence $\mathcal{A}_{\mathsf{mac}}$ only makes permitted queries.

By construction, $\mathcal{A}_{\mathsf{mac}}$ simulates either G_0 or G_1 for $\mathcal{A}$, depending on its secret bit b from game SUFCMRKA. More precisely, it simulates G_0 and G_1 if its own challenge is 1 and 0, respectively. This gives us

$$\begin{aligned}\Big|\Pr[\mathcal{A}^{\mathsf{G}_0} \Rightarrow 1] - \Pr[\mathcal{A}^{\mathsf{G}_1} \Rightarrow 1]\Big| &= \Big|\Pr[\mathcal{A}^{\mathsf{SUFCMRKA}}_{\mathsf{mac}} \Rightarrow 1 \,|\, b=1] - \Pr[\mathcal{A}^{\mathsf{SUFCMRKA}}_{\mathsf{mac}} \Rightarrow 1 \,|\, b=0]\Big| \\ &= \mathbf{Adv}^{\mathsf{SUFCMRKA}}_{\Gamma}(\mathcal{A}_{\mathsf{mac}}, \Phi_m)\,.\end{aligned}$$

For the term $\Pr[\mathcal{A}^{\mathsf{G}_1} \Rightarrow 1] - \Pr[\mathcal{A}^{\mathsf{G}_2} \Rightarrow 1]$, we construct an adversary $\mathcal{A}_{\mathsf{prf}}$ against the RKA-PRF security of the tagging algorithm Tag. First, $\mathcal{A}_{\mathsf{prf}}$ chooses a random key K_e to simulate all encryption related functionalities. Queries to Dec by $\mathcal{A}$ are answered with $\perp$. Queries $(N, A, M, (\phi_e, \phi_m))$ to Enc, are processed as follows. The reduction $\mathcal{A}_{\mathsf{prf}}$ invokes its own oracle F on (N, A, M, ϕ_m) to obtain T, locally computes $C \leftarrow \mathtt{Enc}(\phi_e(K_e), N, M \parallel T)$, and sends C back to $\mathcal{A}$. When $\mathcal{A}$ terminates, $\mathcal{A}_{\mathsf{prf}}$ also terminates and outputs whatever $\mathcal{A}$ does.

We briefly argue that $\mathcal{A}_{\mathsf{prf}}$ never repeats a query to F. By construction, every query (N, A, M, ϕ_m) by $\mathcal{A}_{\mathsf{prf}}$ stems from a query $(N, A, M, \phi_e, \phi_m)$ by $\mathcal{A}$. The only cases that result in a repeating query are (1) $\mathcal{A}$ repeats a query and (2) $\mathcal{A}$ makes two queries which

only differ in ϕ_e. However, both cases are forbidden queries for $\mathcal{A}$. This yields that every output of Tag is a random value.

The adversary $\mathcal{A}_{\mathsf{prf}}$ simulates game G_1 for $\mathcal{A}$ if its own challenge bit b equals 1, while it simulates G_2 for $\mathcal{A}$ if b equals 0. Thus it holds that

$$\begin{aligned}\left|\Pr[\mathcal{A}^{\mathsf{G}_1} \Rightarrow 1] - \Pr[\mathcal{A}^{\mathsf{G}_2} \Rightarrow 1]\right| &= \left|\Pr[\mathcal{A}_{\mathsf{prf}}^{\mathsf{rkaPRF}} \Rightarrow 1 \mid b = 1] - \Pr[\mathcal{A}_{\mathsf{prf}}^{\mathsf{rkaPRF}} \Rightarrow 1 \mid b = 0]\right| \\ &= \mathbf{Adv}_{\Gamma}^{\mathsf{rkaPRF}}(\mathcal{A}_{\mathsf{prf}}, \Phi_m)\,.\end{aligned}$$

We bound the final term $\Pr[\mathcal{A}^{\mathsf{G}_2} \Rightarrow 1] - \Pr[\mathcal{A}^{\mathsf{G}_3} \Rightarrow 1]$ by constructing an adversary $\mathcal{A}_{\mathsf{se}}$ against the IND-CPRKA security of the underlying encryption scheme Σ. At the start, $\mathcal{A}_{\mathsf{se}}$ chooses a random key K_m. Any query to Dec is answered with $\perp$. When $\mathcal{A}$ queries its oracle Enc on $(N, A, M, (\phi_e, \phi_m))$, $\mathcal{A}_{\mathsf{se}}$ chooses a random tag T of length τ, invokes its oracle Enc on $(N, M \parallel T, \phi_e)$ to obtain C, and sends C to $\mathcal{A}$. At the end, $\mathcal{A}_{\mathsf{se}}$ outputs whatever $\mathcal{A}$ outputs.

It holds that $\mathcal{A}_{\mathsf{se}}$ is RKA-nonce-respecting as any query $(N, M \parallel T, \phi_e)$ stems from a query $(N, A, M, \phi_e, \phi_m)$ by $\mathcal{A}$. This means that $\mathcal{A}_{\mathsf{se}}$ repeats a pair of nonce N and RKD function ϕ_e if $\mathcal{A}$ makes two queries using (N, ϕ_e, ϕ_m) and (N, ϕ_e, ϕ'_m). We can distinguish between the cases (1) $\phi_m = \phi'_m$ and (2) $\phi_m \neq \phi'_m$. Case (1) does not occur, as $\mathcal{A}$ is RKA-nonce-respecting and case (2) is forbidden in game $\mathsf{rka\text{-}AE}$. The other option would be that $\mathcal{A}$ makes two queries differing only in the associated data A. This turns out not to be an issue, as the tag T that $\mathcal{A}_{\mathsf{se}}$ queries along with the message depends on A, i.e., different A results in a different message queried by $\mathcal{A}_{\mathsf{se}}$.

The adversary $\mathcal{A}_{\mathsf{se}}$ perfectly simulates games G_2 or G_3 for $\mathcal{A}$ depending on its own challenge from $\mathsf{INDCPRKA}$. Hence, we have

$$\begin{aligned}\left|\Pr[\mathcal{A}^{\mathsf{G}_2} \Rightarrow 1] - \Pr[\mathcal{A}^{\mathsf{G}_3} \Rightarrow 1]\right| &= \left|\Pr[\mathcal{A}_{\mathsf{se}}^{\mathsf{INDCPRKA}} \Rightarrow 1 \mid b = 1] - \Pr[\mathcal{A}_{\mathsf{se}}^{\mathsf{INDCPRKA}} \Rightarrow 1 \mid b = 0]\right| \\ &= \mathbf{Adv}_{\Gamma}^{\mathsf{INDCPRKA}}(\mathcal{A}_{\mathsf{se}}, \Phi_e)\,.\end{aligned}$$

Collecting the bounds above proves the claim. □

Unlike for the N1 and N2 constructions, the s-RKA-AE security of the N3 construction is more subtle. The difference is that the tag is appended to the ciphertext for both the N1 and N2 construction while it is encrypted alongside the message for the N3 construction. The attacks against the N1 and N2 construction rely on the property that the ciphertext consists of two parts which can be manipulated separately. Due to the construction, such attacks do not work against the N3 construction.

It turns out that the s-RKA-AE security of the N3 construction crucially depends on the used encryption scheme. Namely, if the underlying encryption scheme is a stream cipher, then the N3 construction is s-RKA-AE insecure. Below we show an attack against any instantiation using a stream cipher. For such ciphers the ciphertext is the XOR of the message and a keystream derived from the key and the nonce.

Theorem 4.2.6. *Let $\Sigma = (\mathtt{Enc}, \mathtt{Dec})$ be a stream cipher and $\Gamma = (\mathtt{Tag}, \mathtt{Ver})$ be a MAC with RKA function sets Φ_e and Φ_m, respectively. Further, let* N3 *be the AEAD scheme built from Σ and Γ using the N2 construction with RKA function set $\Phi_{ae} = \Phi_e \times \Phi_m$. Then,*

Game G_i

$(K_e, K_m) \leftarrow_\$ \mathcal{K}$
$b' \leftarrow \mathcal{A}^{\mathsf{Enc},\mathsf{Dec}}()$

oracle $\mathsf{Dec}(N, A, C, (\phi_e, \phi_m))$ in G_0

$M \parallel T \leftarrow \mathtt{Dec}(\phi_e(K_e), N, C)$
if $\mathtt{Ver}(\phi_m(K_m), N, A, M, T) = \top$
  return M
return $\bot$

oracle $\mathsf{Dec}(N, A, C, (\phi_e, \phi_m))$ in G_1, G_2, G_3

return $\bot$

oracle $\mathsf{Enc}(N, A, M, (\phi_e, \phi_m))$ in G_0, G_1

$T \leftarrow \mathtt{Tag}(\phi_m(K_m), N, A, M)$
$C \leftarrow \mathtt{Enc}(\phi_e(K_e), N, M \parallel T)$
return C

oracle $\mathsf{Enc}(N, A, M, (\phi_e, \phi_m))$ in G_2

$T \leftarrow_\$ \{0,1\}^\tau$
$C \leftarrow \mathtt{Enc}(\phi_e(K_e), N, M \parallel T)$
return C

oracle $\mathsf{Enc}(N, A, M, (\phi_e, \phi_m))$ in G_3

$C \leftarrow_\$ \{0,1\}^\gamma$
return C

Figure 4.6: Games G_i used to prove Theorem 4.2.5 (RKA-AE security of N3).

N3 *is not* s-RKA-AE*-secure. There exists an RKA-nonce-respecting and* Φ_{ae}*-restricted RKA adversary* $\mathcal{A}$ *such that*

$$\mathbf{Adv}^{\mathsf{s\text{-}rka\text{-}AE}}_{\mathrm{N3}}(\mathcal{A}) = 1 - 2^{-\mu}\,.$$

Proof. Adversary $\mathcal{A}$ chooses a nonce N, associated data A, a message M (of length μ), RKD functions ϕ_e, ϕ_m, and ϕ'_m from the respective sets such that $\phi_m \neq \phi'_m$. Then it queries its encryption oracle Enc on $(N, A, M, (\phi_e, \phi_m))$ and $(N, A, M, (\phi_e, \phi'_m))$ to obtain ciphertext C_1 and C_2. If the first μ bits of C_1 and C_2 are equal, $\mathcal{A}$ outputs 1, otherwise, it outputs 0.

In case $b = 1$, it holds that $C_1 = \mathtt{Enc}(\phi_e(K_e), N, M \parallel \mathtt{Tag}(\phi_m(K_m), N, A, M))$ and $C_2 = \mathtt{Enc}(\phi_e(K_e), N, M \parallel \mathtt{Tag}(\phi'_m(K_m), N, A, M))$. Since the encryption uses the same nonce and the same key, the same keystream for the stream cipher will be used. Together with the fact that the first μ bits are identical as the same message is encrypted, this yields that C_1 and C_2 agree on the first μ bits. Hence

$$\Pr[\mathcal{A}^{\mathsf{s\text{-}rka\text{-}AE}} \Rightarrow 1 \,|\, b = 1] = 1\,.$$

In case $b = 0$, both C_1 and C_2 are chosen at random. The probability that $\mathcal{A}$ outputs 1 is the probability that the first μ bits of C_1 and C_2 are equal, more precisely

$$\Pr[\mathcal{A}^{\mathsf{s\text{-}rka\text{-}AE}} \Rightarrow 1 \,|\, b = 0] = 2^{-\mu}\,.$$

Collecting the above yields

$$\begin{aligned}\mathbf{Adv}^{\mathsf{s\text{-}rka\text{-}AE}}_{\mathrm{N1}}(\mathcal{A}) &= \left|\Pr[\mathcal{A}^{\mathsf{s\text{-}rka\text{-}AE}} \Rightarrow 1 \,|\, b = 1] - \Pr[\mathcal{A}^{\mathsf{s\text{-}rka\text{-}AE}} \Rightarrow 1 \,|\, b = 0]\right| \\ &= 1 - 2^{-\mu}\,.\end{aligned}$$

This concludes the proof. □

In the attack above, the RKA-nonce-respecting adversary essentially bypasses the nonce-respecting property of the underlying encryption scheme by repeating the nonce N and the RKD function ϕ_e for the encryption scheme. Then it exploits the fact that the underlying stream cipher is secure only against nonce-respecting adversaries. We conjecture that any instantiation using an encryption scheme that can be broken in the nonce-misuse case results in an s-RKA-AE insecure instantiation of the N3 construction. The problematic part is that both the message and the tag are encrypted. While the adversary has full control over the former, it cannot choose the latter at will. This seems to thwart a simple proof showing that any nonce-misuse adversary against the underlying encryption scheme can be turned into an s-RKA-AE adversary against N3.

4.2.4 Related-Key Attack Secure Encryption

In this section we show that the encryption scheme $\textsc{Se}[\mathcal{F}, \mathcal{G}]$ proposed in Chapter 3 is RKA-secure. The scheme comprises a function $\mathcal{F}$ and a pseudorandom generator $\mathcal{G}$ and is displayed in Figure 4.7. The ciphertext is computed by feeding the nonce N into the function $\mathcal{F}$, expanding the output using the PRG $\mathcal{G}$, and XORing the output to the message. The scheme achieves IND-CPRKA security against RKA-nonce-respecting and Φ-restricted adversaries, where Φ is specified from the RKA-PRF security of the underlying function $\mathcal{F}$. The result is given in the following theorem.

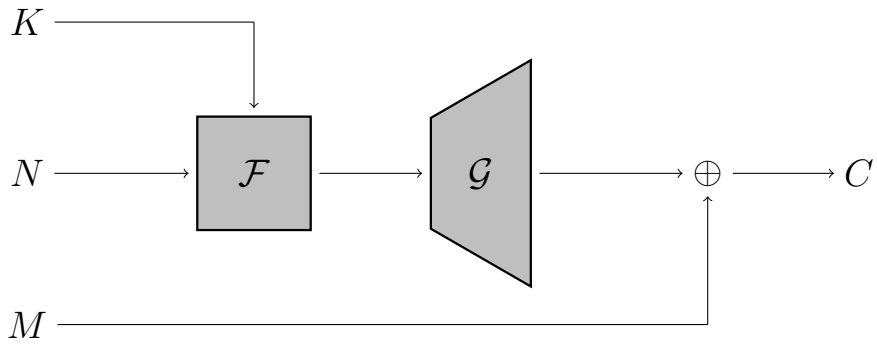

Figure 4.7: Visualisation of the encryption scheme $\textsc{Se}[\mathcal{F}, \mathcal{G}]$.

Theorem 4.2.7. *Let $\Sigma = (\mathsf{Enc}, \mathsf{Dec})$ be an encryption scheme built from a function $\mathcal{F} : \mathcal{K} \times \{0,1\}^\nu \to \{0,1\}^n$ and a pseudorandom generator $\mathcal{G} : \{0,1\}^n \to \{0,1\}^\mu$ as displayed in Figure 4.7. Then, for any RKA-nonce-respecting and Φ-restricted RKA adversary $\mathcal{A}$, making q queries to Enc, that never repeats a query, there exists a Φ-restricted RKA adversary $\mathcal{A}_{\mathsf{prf}}$ and an adversary $\mathcal{A}_{\mathsf{prg}}$ such that*

$$\mathbf{Adv}^{\mathsf{INDCPRKA}}_{\Sigma}(\mathcal{A}) \leq \mathbf{Adv}^{\mathsf{rkaPRF}}_{\mathcal{F}}(\mathcal{A}_{\mathsf{prf}}) + q\,\mathbf{Adv}^{\mathsf{PRG}}_{\mathcal{G}}(\mathcal{A}_{\mathsf{prg}})\,.$$

Proof. The proof uses three games G_0, G_1, and G_2 displayed in Figure 4.8. It holds that game G_0 equals game $\mathsf{INDCPRKA}$ with secret bit fixed to 1, while game G_2 equals game

INDCPRKA with secret bit fixed to 0, hence

$$\begin{aligned}\mathbf{Adv}^{\mathsf{INDCPRKA}}_{\Sigma}(\mathcal{A},\Phi) &= \left|\Pr[\mathcal{A}^{\mathsf{INDCPRKA}} \Rightarrow 1 \mid b=1] - \Pr[\mathcal{A}^{\mathsf{INDCPRKA}} \Rightarrow 1 \mid b=0]\right| \\ &= \left|\Pr[\mathcal{A}^{\mathsf{G}_0} \Rightarrow 1] - \Pr[\mathcal{A}^{\mathsf{G}_2} \Rightarrow 1]\right| \\ &\leq \left|\Pr[\mathcal{A}^{\mathsf{G}_0} \Rightarrow 1] - \Pr[\mathcal{A}^{\mathsf{G}_1} \Rightarrow 1]\right| + \left|\Pr[\mathcal{A}^{\mathsf{G}_1} \Rightarrow 1] - \Pr[\mathcal{A}^{\mathsf{G}_2} \Rightarrow 1]\right| .\end{aligned}$$

For the first difference we construct the following adversary $\mathcal{A}_{\mathsf{prf}}$ against the RKA-PRF security of $\mathcal{F}$ playing game rkaPRF. It runs $\mathcal{A}$ and processes every query (N, M, ϕ) as follows. It queries (N, ϕ) to its own oracle F and the response is used as input to $\mathcal{G}$. The ciphertext C is obtained by XORing the output of $\mathcal{G}$ to the message M and C is sent back to $\mathcal{A}$. When $\mathcal{A}$ outputs a bit b', $\mathcal{A}_{\mathsf{prf}}$ outputs b' as well.

If the secret bit in game rkaPRF is 1, $\mathcal{A}_{\mathsf{prf}}$ has access to $\mathcal{F}$, hence it perfectly simulates game G_0 for $\mathcal{A}$. If, on the other hand, the secret bit in game rkaPRF is 0, $\mathcal{A}_{\mathsf{prf}}$ has access to a random function $\mathcal{F}^*$ instead and perfectly simulates game G_1 for $\mathcal{A}$. By outputting the same as $\mathcal{A}$, it holds that

$$\begin{aligned}\Pr[\mathcal{A}^{\mathsf{G}_0} \Rightarrow 1] - \Pr[\mathcal{A}^{\mathsf{G}_1} \Rightarrow 1] &= \left|\Pr[\mathcal{A}^{\mathsf{rkaPRF}}_{\mathsf{prf}} \Rightarrow 1 \mid b=1] - \Pr[\mathcal{A}^{\mathsf{rkaPRF}}_{\mathsf{prf}} \Rightarrow 1 \mid b=0]\right| \\ &= \mathbf{Adv}^{\mathsf{rkaPRF}}_{\mathcal{F}}(\mathcal{A}_{\mathsf{prf}},\Phi) .\end{aligned}$$

For the second difference we introduce the hybrid games $\mathsf{H}_1, \ldots, \mathsf{H}_q$ displayed in Figure 4.9. Game H_0 equals game G_1 as all queries are answered using $\mathcal{G}$ to obtain the bit string that is XORed to the message. Likewise, H_q equals G_2 as all queries are answered by sampling a random bit string that is then XORed to the message. Hence, we obtain

$$\begin{aligned}\Pr[\mathcal{A}^{\mathsf{G}_1} \Rightarrow 1] - \Pr[\mathcal{A}^{\mathsf{G}_2} \Rightarrow 1] &= \left|\Pr[\mathcal{A}^{\mathsf{H}_0} \Rightarrow 1] - \Pr[\mathcal{A}^{\mathsf{H}_q} \Rightarrow 1]\right| \\ &\leq \sum_{i=1}^{q} \left|\Pr[\mathcal{A}^{\mathsf{H}_{i-1}} \Rightarrow 1] - \Pr[\mathcal{A}^{\mathsf{H}_i} \Rightarrow 1]\right| .\end{aligned}$$

Let $\mathcal{A}_{\mathsf{i}}$ be the following adversary against $\mathcal{G}$ that receives a challenge Z which is either the output of $\mathcal{G}$ on a randomly chosen seed or a random bit string. It runs $\mathcal{A}$, answering the first $i-1$ queries by sampling a random bit string of length μ, XORing it to the queried message to get the ciphertext which it sends back to $\mathcal{A}$. For the i^{th} query (N, M), $\mathcal{A}_{\mathsf{i}}$ computes $C \leftarrow Z \oplus M$ and sends C back to $\mathcal{A}$. For the remaining $q-i$ queries, $\mathcal{A}_{\mathsf{i}}$ samples a seed z at random, compute $C \leftarrow \mathcal{G}(z) \oplus M$, and sends C back to $\mathcal{A}$. When $\mathcal{A}$ terminates and output a bit b', $\mathcal{A}_{\mathsf{i}}$ forwards it as its own output.

It holds that $\mathcal{A}_{\mathsf{i}}$ simulates game H_{i-1} if its own challenge is the output of $\mathcal{G}$ on a random seed and H_i else. By outputting the same as $\mathcal{A}$, we get

$$\begin{aligned}\left|\Pr[\mathcal{A}^{\mathsf{H}_{i-1}} \Rightarrow 1] - \Pr[\mathcal{A}^{\mathsf{H}_i} \Rightarrow 1]\right| &= \left|\Pr[\mathcal{A}^{\mathsf{PRG}}_{\mathsf{i}} \Rightarrow 1 \mid b=1] - \Pr[\mathcal{A}^{\mathsf{PRG}}_{\mathsf{i}} \Rightarrow 1 \mid b=0]\right| \\ &= \mathbf{Adv}^{\mathsf{PRG}}_{\mathcal{G}}(\mathcal{A}_{\mathsf{i}}) .\end{aligned}$$

By standard hybrid argument [FM21], we define $\mathcal{A}_{\mathsf{prg}}$ as the adversary that picks $i \leftarrow\!\!\$\, [q]$ and then behaves as $\mathcal{A}_{\mathsf{i}}$ and obtain

$$\begin{aligned}\left|\Pr[\mathcal{A}^{\mathsf{G}_1} \Rightarrow 1] - \Pr[\mathcal{A}^{\mathsf{G}_2} \Rightarrow 1]\right| &\leq \sum_{i=1}^{q} \left|\Pr[\mathcal{A}^{\mathsf{H}_{i-1}} \Rightarrow 1] - \Pr[\mathcal{A}^{\mathsf{H}_i} \Rightarrow 1]\right| \\ &\leq q\, \mathbf{Adv}^{\mathsf{PRG}}_{\mathcal{G}}(\mathcal{A}_{\mathsf{prg}})\,.\end{aligned}$$

Combining the above finally yields

$$\begin{aligned}\mathbf{Adv}^{\mathsf{INDCPRKA}}_{\Sigma}(\mathcal{A}, \Phi) &\leq \left|\Pr[\mathcal{A}^{\mathsf{G}_0} \Rightarrow 1] - \Pr[\mathcal{A}^{\mathsf{G}_1} \Rightarrow 1]\right| + \left|\Pr[\mathcal{A}^{\mathsf{G}_1} \Rightarrow 1] - \Pr[\mathcal{A}^{\mathsf{G}_2} \Rightarrow 1]\right| \\ &\leq \mathbf{Adv}^{\mathsf{rkaPRF}}_{\mathcal{F}}(\mathcal{A}_{\mathsf{prf}}, \Phi) + q\, \mathbf{Adv}^{\mathsf{PRG}}_{\mathcal{G}}(\mathcal{A}_{\mathsf{prg}})\,.\end{aligned}$$

This concludes the proof. □

Game G_i

$K \leftarrow\!\!\$\, \mathcal{K}$
$\mathcal{F}^* \leftarrow\!\!\$\, \mathsf{Func}(\mathcal{K}, \mathcal{N}, \{0,1\}^y)$
$b' \leftarrow \mathcal{A}^{\mathsf{Enc}}()$

$\mathsf{Enc}(N, M, \phi)$ in G_0

$C \leftarrow \mathcal{G}(\mathcal{F}(\phi(K), N)) \oplus M$
return C

$\mathsf{Enc}(N, M, \phi)$ in G_1

$C \leftarrow \mathcal{G}(\mathcal{F}^*(\phi(K), N)) \oplus M$
return C

$\mathsf{Enc}(N, M, \phi)$ in G_2

$Z \leftarrow\!\!\$\, \{0,1\}^\mu$
$C \leftarrow Z \oplus M$
return C

Figure 4.8: Games G_0, G_1, and G_2 used to prove the INDCPRKA security of Theorem 4.2.7.

Game H_i

$c \leftarrow 0$
$\mathcal{F}^* \leftarrow\!\!\$\, \mathsf{Func}(\mathcal{K}, \mathcal{N}, \{0,1\}^y)$
$b' \leftarrow \mathcal{A}^{\mathsf{Enc}}()$

$\mathsf{Enc}(N, M, \phi)$ in H_i

$c \leftarrow c + 1$
if $c \leq i$
  $z \leftarrow \mathcal{F}^*(\phi(K), N)$
  $C \leftarrow \mathcal{G}(z) \oplus M$
else
  $Z \leftarrow\!\!\$\, \{0,1\}^\mu$
  $C \leftarrow Z \oplus M$
return C

Figure 4.9: Hybrid games H_i used to prove the INDCPRKA security of Theorem 4.2.7.

4.2.5 Related-Key Attack Secure Message Authentication

In this section we show that the message authentication code $\textsc{Mac}[\mathcal{H}, \mathcal{F}']$ proposed in Chapter 3 is RKA-secure. The scheme comprises a hash function $\mathcal{H}$ and a function $\mathcal{F}'$ and is displayed in Figure 4.10. The tag is obtained by evaluating the hash function $\mathcal{H}$ on the message M and evaluating the function $\mathcal{F}'$ on the hash value.

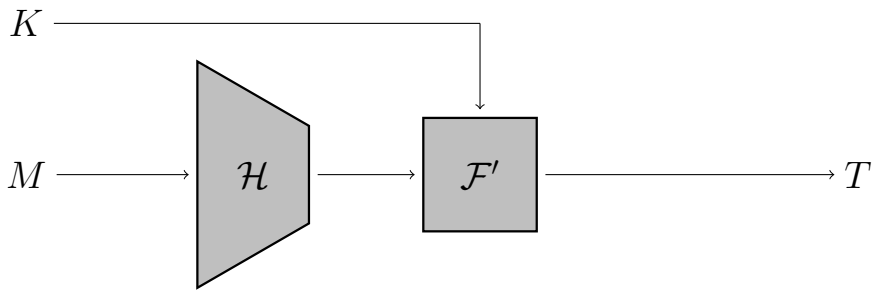

Figure 4.10: Visualisation of the message authentication code $\textsc{Mac}[\mathcal{H}, \mathcal{F}']$.

RKA-Secure MAC from RKA-secure PRF

The theorem below shows that an RKA-secure pseudorandom function yields an RKA-secure MAC via the canonical MAC construction. That is, the tag is computed by evaluating the function on the message and verification works by recomputing the tag and comparison with the candidate tag.

Theorem 4.2.8 (RKA-PRF yields RKA-MAC)**.** *Let $\mathcal{F}' \colon \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ be a function and Γ be the canonical MAC built from $\mathcal{F}'$. Then, for any Φ-restricted RKA adversary $\mathcal{A}$ against Γ playing* SUFCMRKA*, making q queries to* Vfy*, there exists a Φ-restricted RKA adversary $\mathcal{A}_{\mathsf{prf}}$ against $\mathcal{F}'$ playing* rkaPRF *such that*

$$\mathbf{Adv}^{\mathsf{SUFCMRKA}}_{\Gamma}(\mathcal{A}, \Phi) \leq 2\,\mathbf{Adv}^{\mathsf{rkaPRF}}_{\mathcal{F}'}(\mathcal{A}_{\mathsf{prf}}, \Phi) + \frac{q}{|\mathcal{Y}|}\,.$$

Proof. We prove the theorem using the games G_0, G_1, G_2, and G_3 displayed in Figure 4.11. Game G_0 is SUFCMRKA instantiated with $\mathcal{F}'$ and secret bit $b = 1$. In game G_1, both oracles Tag and Vfy use a random function to generate the tags. In game G_2, oracle Vfy rejects any queried tag. In game G_3, oracle Tag uses again $\mathcal{F}'$ instead of choosing the tags at random, thus it corresponds to SUFCMRKA with secret bit $b = 0$. It holds that

$$\begin{aligned}
\mathbf{Adv}^{\mathsf{SUFCMRKA}}_{\Gamma}(\mathcal{A}, \Phi) &= \Big|\Pr[\mathcal{A}^{\mathsf{SUFCMRKA}} \Rightarrow 1 \,|\, b = 1] - \Pr[\mathcal{A}^{\mathsf{SUFCMRKA}} \Rightarrow 1 \,|\, b = 0]\Big| \\
&= \Big|\Pr[\mathcal{A}^{\mathsf{G}_0} \Rightarrow 1] - \Pr[\mathcal{A}^{\mathsf{G}_3} \Rightarrow 1]\Big| \\
&\leq \sum_{i=1}^{3} \Big|\Pr[\mathcal{A}^{\mathsf{G}_{i-1}} \Rightarrow 1] - \Pr[\mathcal{A}^{\mathsf{G}_i} \Rightarrow 1]\Big|\,.
\end{aligned}$$

We transform any adversary $\mathcal{A}$ that distinguishes between G_0 and G_1 into an adversary $\mathcal{A}_0$ against $\mathcal{F}'$. For each query (X, ϕ) to Tag, $\mathcal{A}_0$ queries its oracle F on (X, ϕ) and sends

the response T back to $\mathcal{A}$. For each query (X, T, ϕ) to Vfy, $\mathcal{A}_0$ queries its oracle F on (X, ϕ) to get T'. If $T = T'$ $\mathcal{A}_0$ sends $\top$ back to $\mathcal{A}$, otherwise, it sends $\perp$ back. When $\mathcal{A}$ eventually outputs a bit b', $\mathcal{A}_0$ outputs the same.

It holds that $\mathcal{A}_0$ perfectly simulates game G_0 and game G_1 when its own challenge bit b, from game PRF, is 1 and 0, respectively. We have

$$\begin{aligned}\Big|\Pr[\mathcal{A}^{\mathsf{G}_0} \Rightarrow 1] - \Pr[\mathcal{A}^{\mathsf{G}_1} \Rightarrow 1]\Big| &= \Big|\Pr[\mathcal{A}_0^{\mathsf{rkaPRF}} \Rightarrow 1 \mid b = 1] - \Pr[\mathcal{A}_0^{\mathsf{rkaPRF}} \Rightarrow 1 \mid b = 0]\Big| \\ &= \mathbf{Adv}^{\mathsf{rkaPRF}}_{\mathcal{F}'}(\mathcal{A}_0, \Phi)\,.\end{aligned}$$

The advantage in distinguishing between G_1 and G_2 is a simple counting argument. The adversary can only distinguish if it guesses the output of the random function, in which case Vfy returns $\top$ in G_1 and $\perp$ in G_2. Since $\mathcal{A}$ makes q queries to Vfy we get

$$\Big|\Pr[\mathcal{A}^{\mathsf{G}_1} \Rightarrow 1] - \Pr[\mathcal{A}^{\mathsf{G}_2} \Rightarrow 1]\Big| \leq \frac{q}{|\mathcal{Y}|}\,.$$

Distinguishing G_2 and G_3 essentially asks to distinguish where oracle Tag is implemented using $\mathcal{F}'$ or a random function. We construct the following adversary $\mathcal{A}_2$. It returns $\perp$ for every query that $\mathcal{A}$ makes to Vfy. For queries (X, ϕ) to Tag, $\mathcal{A}_2$ forwards the query to its own oracle F to obtain T which it sends to $\mathcal{A}$. When $\mathcal{A}$ outputs a bit b', $\mathcal{A}_2$ outputs $1 - b'$.

It holds that $\mathcal{A}_2$ perfectly simulates game G_2 if its own challenge bit b equals 0 while it simulates G_3 if b equals 1. Thus we have

$$\begin{aligned}\Big|\Pr[\mathcal{A}^{\mathsf{G}_2} \Rightarrow 1] - \Pr[\mathcal{A}^{\mathsf{G}_3} \Rightarrow 1]\Big| &= \Big|\Pr[\mathcal{A}_2^{\mathsf{rkaPRF}} \Rightarrow 0 \mid b = 0] - \Pr[\mathcal{A}_2^{\mathsf{rkaPRF}} \Rightarrow 0 \mid b = 1]\Big| \\ &= \mathbf{Adv}^{\mathsf{rkaPRF}}_{\mathcal{F}'}(\mathcal{A}_2, \Phi)\,.\end{aligned}$$

Collecting the bounds and defining $\mathcal{A}_{\mathsf{prf}}$ to be the adversary with the higher advantage among $\mathcal{A}_0$ and $\mathcal{A}_2$ gives the desired result. □

Games $\mathsf{G}_0, \mathsf{G}_1, \mathsf{G}_2, \mathsf{G}_3$

$K \leftarrow_{\$} \mathcal{K}$
$\mathcal{F}^* \leftarrow_{\$} \mathsf{Func}(\mathcal{K}, \mathcal{X}, \mathcal{Y})$
$b' \leftarrow \mathcal{A}^{\mathsf{Tag},\mathsf{Vfy}}()$

oracle $\mathsf{Vfy}(M, T, \phi)$ in G_0

$T' \leftarrow \mathcal{F}'(\phi(K), M)$
return $(T' = T)$

oracle $\mathsf{Vfy}(M, T, \phi)$ in G_1

$T' \leftarrow_{\$} \mathcal{F}^*(\phi(K), M)$
return $(T' = T)$

oracle $\mathsf{Vfy}(M, T, \phi)$ in G_2, G_3

return $\perp$

oracle $\mathsf{Tag}(M, \phi)$ in G_0, G_3

$T \leftarrow \mathcal{F}'(\phi(K), M)$
return T

oracle $\mathsf{Tag}(M, \phi)$ in G_1, G_2

$T \leftarrow_{\$} \mathcal{F}^*(\phi(K), M)$
return T

Figure 4.11: Games used in the proof of Theorem 4.2.8.

RKA-Secure PRF from RKA-secure PRF and Hash

Theorem 4.2.8 shows that we can construct a RKA-secure MAC from a RKA-secure pseudorandom function. The main restriction is that the message space of the MAC equals the message space of the function. The following theorem shows that this is not a hindrance. It shows that the message space of a RKA-secure pseudorandom function can be extended to arbitrary long messages by first hashing the input and then applying the function to the hash value.

Theorem 4.2.9 (Hash and RKA-PRF yields RKA-PRF)**.** *Let $\mathcal{F}\colon \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ be a function, $\mathcal{H}\colon \{0,1\}^* \to \mathcal{X}$ be a hash function and $\mathcal{F}'\colon \{0,1\}^* \to \mathcal{Y}, X \mapsto \mathcal{F}(\mathcal{H}(X))$ be a function. Then for any Φ-restricted RKA adversary $\mathcal{A}$ against $\mathcal{F}'$ playing* rkaPRF*, there exists a Φ-restricted RKA adversary $\mathcal{A}_{\mathsf{prf}}$ against $\mathcal{F}$ playing* rkaPRF *and an adversary $\mathcal{A}_{\mathsf{hash}}$ against $\mathcal{H}$ playing* CR *such that*

$$\mathbf{Adv}^{\mathsf{rkaPRF}}_{\mathcal{F}'}(\mathcal{A}, \Phi) \leq 2\,\mathbf{Adv}^{\mathsf{CR}}_{\mathcal{H}}(\mathcal{A}_{\mathsf{hash}}) + \mathbf{Adv}^{\mathsf{rkaPRF}}_{\mathcal{F}}(\mathcal{A}_{\mathsf{prf}}, \Phi)\,.$$

Proof. We use the games G_0, $\boxed{\mathsf{G}_0}$, $\boxed{\mathsf{G}_1}$, and G_1 (cf. Figure 4.12) to prove the statement. Game G_0 corresponds to game rkaPRF with secret bit b fixed to 1. Likewise, G_1 corresponds to rkaPRF with b fixed to 0. The boxed versions differ in that the oracle returns $\perp$ if the hash value occurs more than once. Thus, it holds that

$$\begin{aligned}
\mathbf{Adv}^{\mathsf{rkaPRF}}_{\mathcal{F}'}(\mathcal{A}, \Phi) &= \left|2\ \Pr[\mathsf{rkaPRF}^{\mathcal{A}} \Rightarrow 1] - 1\right| \\
&= \left|\Pr[\mathcal{A}^{\mathsf{rkaPRF}} \Rightarrow 1 \mid b = 1] - \Pr[\mathcal{A}^{\mathsf{rkaPRF}} \Rightarrow 1 \mid b = 0]\right| \\
&= \left|\Pr[\mathcal{A}^{\mathsf{G}_0} \Rightarrow 1] - \Pr[\mathcal{A}^{\mathsf{G}_1} \Rightarrow 1]\right| \\
&\leq \left|\Pr[\mathcal{A}^{\mathsf{G}_0} \Rightarrow 1] - \Pr[\mathcal{A}^{\boxed{\mathsf{G}_0}} \Rightarrow 1] \Rightarrow 1]\right| \\
&\quad + \left|\Pr[\mathcal{A}^{\boxed{\mathsf{G}_0}} - \Pr[\mathcal{A}^{\boxed{\mathsf{G}_1}} \Rightarrow 1]\right| \\
&\quad + \left|\Pr[\mathcal{A}^{\boxed{\mathsf{G}_1}} \Rightarrow 1] - \Pr[\mathcal{A}^{\mathsf{G}_1} \Rightarrow 1]\right|\,.
\end{aligned}$$

Games G_0 and $\boxed{\mathsf{G}_0}$ (resp. $\boxed{\mathsf{G}_1}$ and G_1) are identical-until-bad games, hence

$$\left|\Pr[\mathcal{A}^{\mathsf{G}_0} \Rightarrow 1] - \Pr[\mathcal{A}^{\boxed{\mathsf{G}_0}} \Rightarrow 1]\right| \leq \Pr[\mathcal{A}^{\mathsf{G}_0} \text{ sets } \mathsf{Bad}]$$

and

$$\left|\Pr[\mathcal{A}^{\boxed{\mathsf{G}_1}} \Rightarrow 1] - \Pr[\mathcal{A}^{\mathsf{G}_1} \Rightarrow 1]\right| \leq \Pr[\mathcal{A}^{\mathsf{G}_1} \text{ sets } \mathsf{Bad}]\,.$$

We construct an adversary $\mathcal{A}_0$ to bound the probability that Bad is set in G_0. Adversary $\mathcal{A}_0$ first chooses a random key K and runs $\mathcal{A}$. For every query (X, ϕ) by $\mathcal{A}$, $\mathcal{A}_0$ computes the hash value $H \leftarrow \mathcal{H}(X)$ and checks whether H has been recorded in the set $\mathcal{H}$. If so, $\mathcal{A}$ outputs X and the colliding input as collision for the hash function $\mathcal{H}$. If the input is not colliding, $\mathcal{A}_0$ computes $T \leftarrow \mathcal{F}(\phi(K), H)$ and sends y back to $\mathcal{A}$. Adversary $\mathcal{A}_1$ to bound the probability that Bad is set in game G_1 is constructed very similar. The mere

difference is that $\mathcal{A}_1$ does not need a key and simply returns random values T, using lazy sampling, to $\mathcal{A}$. Thus, we have

$$\left|\Pr[\mathcal{A}^{\mathsf{G}_0} \Rightarrow 1] - \Pr[\mathcal{A}^{\boxed{\mathsf{G}_0}} \Rightarrow 1]\right| \leq \Pr[\mathcal{A}^{\mathsf{G}_0} \text{ sets } \mathsf{Bad}] \leq \mathbf{Adv}^{\mathsf{CR}}_{\mathcal{H}}(\mathcal{A}_0)$$

and

$$\left|\Pr[\mathcal{A}^{\boxed{\mathsf{G}_1}} \Rightarrow 1] - \Pr[\mathcal{A}^{\mathsf{G}_1} \Rightarrow 1]\right| \leq \Pr[\mathcal{A}^{\mathsf{G}_1} \text{ sets } \mathsf{Bad}] \leq \mathbf{Adv}^{\mathsf{CR}}_{\mathcal{H}}(\mathcal{A}_1)\,.$$

It remains to bound the advantage of $\mathcal{A}$ in distinguishing game $\boxed{\mathsf{G}_0}$ and $\boxed{\mathsf{G}_1}$. We construct the following adversary $\mathcal{A}_{\mathsf{prf}}$. For every query (X, ϕ) by $\mathcal{A}$, $\mathcal{A}_{\mathsf{prf}}$ first computes $H \leftarrow \mathcal{H}(X)$ and returns $\perp$ if H is recorded in the set $\mathcal{H}$. Otherwise, it records H in the set $\mathcal{H}$ and queries its own oracle F on (H, ϕ) and sends the response back to $\mathcal{A}$. Finally, $\mathcal{A}_{\mathsf{prf}}$ outputs whatever $\mathcal{A}$ outputs.

It holds that $\mathcal{A}_{\mathsf{prf}}$ perfectly simulates game $\boxed{\mathsf{G}_0}$ if its own challenge bit b is 1 and game $\boxed{\mathsf{G}_1}$ if it is 0. By outputting the same as $\mathcal{A}$, we get

$$\begin{aligned}\left|\Pr[\mathcal{A}^{\boxed{\mathsf{G}_0}} \Rightarrow 1] - \Pr[\mathcal{A}^{\boxed{\mathsf{G}_1}} \Rightarrow 1]\right| &= \left|\Pr[\mathcal{A}^{\mathsf{rkaPRF}}_{\mathsf{prf}} \Rightarrow 1 \,|\, b = 1] - \Pr[\mathcal{A}^{\mathsf{rkaPRF}}_{\mathsf{prf}} \Rightarrow 1 \,|\, b = 0]\right| \\ &= \mathbf{Adv}^{\mathsf{rkaPRF}}_{\mathcal{F}}(\mathcal{A}_{\mathsf{prf}}, \Phi)\,.\end{aligned}$$

Collecting the bounds and letting $\mathcal{A}_{\mathsf{hash}}$ be the adversary with the higher advantage among $\mathcal{A}_0$ and $\mathcal{A}_1$ gives

$$\begin{aligned}\mathbf{Adv}^{\mathsf{rkaPRF}}_{\mathcal{F}'}(\mathcal{A}, \Phi) &\leq \left|\Pr[\mathcal{A}^{\mathsf{G}_0} \Rightarrow 1] - \Pr[\mathcal{A}^{\boxed{\mathsf{G}_0}} \Rightarrow 1]\right| \\ &\quad + \left|\Pr[\mathcal{A}^{\boxed{\mathsf{G}_0}} \Rightarrow 1] - \Pr[\mathcal{A}^{\boxed{\mathsf{G}_1}} \Rightarrow 1]\right| \\ &\quad + \left|\Pr[\mathcal{A}^{\boxed{\mathsf{G}_1}} \Rightarrow 1] - \Pr[\mathcal{A}^{\mathsf{G}_1} \Rightarrow 1]\right| \\ &\leq \mathbf{Adv}^{\mathsf{CR}}_{\mathcal{H}}(\mathcal{A}_0) + \mathbf{Adv}^{\mathsf{rkaPRF}}_{\mathcal{F}}(\mathcal{A}_{\mathsf{prf}}, \Phi) + \mathbf{Adv}^{\mathsf{CR}}_{\mathcal{H}}(\mathcal{A}_1) \\ &\leq 2\,\mathbf{Adv}^{\mathsf{CR}}_{\mathcal{H}}(\mathcal{A}_{\mathsf{hash}}) + \mathbf{Adv}^{\mathsf{rkaPRF}}_{\mathcal{F}}(\mathcal{A}_{\mathsf{prf}}, \Phi)\,.\end{aligned}$$

This proves the claim. □

4.3 Public Key Encryption Vulnerable to Resetting Attacks

To show that the LoR-IND-CPRA security notion (cf. Definition 2.5.3) is strictly stronger than the classical notion of ciphertext indistinguishability (IND-CPA), Yilek [Yil10b] gives a separation example, i.e., a PKE scheme that is IND-CPA-secure but LoR-IND-CPRA-insecure. The scheme follows the standard hybrid encryption idea and combines an arbitrary public key encryption scheme with the one-time pad encryption.[20] The concrete scheme[21] is displayed in Figure 4.13. The core observation is that in the resetting attack model, the same one-time key will be used for every query as it is derived from the (reused)

[20] We note that [Yil10a] shows that the ElGamal PKE scheme [ElG85] is LoR-IND-CPRA insecure as well.
[21] In [Yil10b] the scheme also consists of a MAC to achieve CCA security which we omit here for simplicity.

Game G_0, $\boxed{\mathsf{G}_0}$, G_1, $\boxed{\mathsf{G}_1}$	**oracle** $\mathsf{F}(X, \phi)$ in G_0, $\boxed{\mathsf{G}_0}$	**oracle** $\mathsf{F}(X, \phi)$ in G_1, $\boxed{\mathsf{G}_1}$
$\mathsf{Bad} \leftarrow \text{false}$	$H \leftarrow \mathcal{H}(X)$	$H \leftarrow \mathcal{H}(X)$
$\mathcal{H} \leftarrow \emptyset$	**if** $H \in \mathcal{H}$	**if** $H \in \mathcal{H}$
$\mathcal{F}^* \leftarrow_\$ \mathsf{Func}(\mathcal{K}, \mathcal{X}, \mathcal{Y})$	$\mathsf{Bad} \leftarrow \text{true}$	$\mathsf{Bad} \leftarrow \text{true}$
$K \leftarrow_\$ \mathcal{K}$	$\boxed{\textbf{return } \perp}$	$\boxed{\textbf{return } \perp}$
$b' \leftarrow \mathcal{A}^{\mathsf{F}}()$	$\mathcal{H} \leftarrow_\cup \{H\}$	$\mathcal{H} \leftarrow_\cup \{H\}$
	$T' \leftarrow \mathcal{F}(\phi(K), H)$	$T' \leftarrow \mathcal{F}^*(\phi(K), M)$
	return T'	**return** T'

Figure 4.12: Games used in the proof of Theorem 4.2.9. Games $\boxed{\mathsf{G}_i}$ contain the boxed code, games G_i do not.

randomness. By making one query to the oracle Enc, the adversary learns this one-time key, which it can then use to decrypt its challenge query. More precisely, the adversary $\mathcal{A}$ queries the message $m = 0 \ldots 0$ to Enc. For the received ciphertext $c = (c_1, c_2)$, it holds that $c_1 = m \oplus k = 0 \ldots 0 \oplus k = k$, hence $\mathcal{A}$ knows the symmetric key k. Then $\mathcal{A}$ can query $\mathsf{LR\text{-}Enc}$ on two randomly chosen messages and decrypt ciphertext part c_1 using k to perfectly determine the secret bit b.

$\mathtt{KGen}(\lambda)$	$\mathtt{Enc}(pk, m; r)$
$(pk, sk) \leftarrow \mathtt{KGen}^P(\lambda)$	$k, r^* \leftarrow r$
return (sk, pk)	$c_1 \leftarrow k \oplus m$
	$c_2 \leftarrow \mathtt{Enc}^P(pk, k; r^*)$
	return $c \leftarrow (c_1, c_2)$

Figure 4.13: Separation example given in [Yil10b]. Algorithms $\mathtt{KGen}^P$ and $\mathtt{Enc}^P$ are the key generation and encryption algorithm of the underlying PKE scheme, respectively.

This clearly shows that the LoR-IND-CPRA security notion is stronger than the classical IND-CPA security notion. However, the attack does not exploit a weakness in the scheme. It essentially bypasses the security by using the one-time pad in an insecure way, namely, using a key twice. We emphasise that this specific attack no longer works if the one-time pad encryption is replaced with a secret key encryption scheme for which using the same secret key does not affect the security. Furthermore, the idea behind the hybrid encryption scheme is to avoid encrypting a large message using a (costly) public key encryption. Since a key for the one-time pad has the same length as the message, instantiating the hybrid encryption scheme with the one-time pad defeats its main advantage. The given separation is therefore more of theoretical interest and raises the question how critical resetting attacks are in practice.

In this section, we show that resetting attacks are devastating in practice by showing that many PKE schemes are susceptible to these attacks. To this end, in Section 4.3.1, we define a class of public key encryption schemes that we call PK-splittable and show that

such schemes are LoR-IND-CPRA-insecure. We then show, in Section 4.3.2, that every PKE scheme following the LWE-based scheme by Regev [Reg05, Reg09], several code-based encryption schemes, and any instantiation of the hybrid encryption scheme—not just the one using the one-time pad—lie in this class of encryption schemes. Hence, all these schemes are insecure against resetting attacks.

4.3.1 A Class of LoR-IND-CPRA-Insecure PKE Schemes

We define the term PK-splittable for public key encryption schemes. Intuitively, these are schemes for which the public key and the ciphertext can be divided into two parts such that: (1) each part of the public key affects exactly one part of the ciphertext and (2) only one part of the ciphertext depends on the message that is encrypted. Below we give the formal definition.

Definition 4.3.1. *Let* $\textsc{Pke} = (\mathtt{KGen}, \mathtt{Enc}, \mathtt{Dec})$ *be a public key encryption scheme, where* $\mathcal{PK} = \mathcal{PK}_g \times \mathcal{PK}_f$ *with* $\mathcal{PK}_g \neq \emptyset$ *and* $\mathcal{C} = \mathcal{X} \times \mathcal{Y}$*. If there exist functions* $f\colon \mathcal{PK}_f \times \mathcal{R} \times \mathcal{M} \to \mathcal{X}$ *and* $g\colon \mathcal{PK}_g \times \mathcal{R} \to \mathcal{Y}$ *such that for any public key* $pk = (pk_g, pk_f)$ *it holds that*

$$\mathtt{Enc}(pk, m; r) = (f(pk_f, r, m), g(pk_g, r))\,,$$

then we say that Pke *is a* PK-splittable *public key encryption scheme with* core encryption function f *and* auxiliary encryption function g*.*

From Definition 4.3.1 it is easy to see that PK-splittable public key encryption schemes are LoR-IND-CPRA insecure. First, the adversary makes a query to the challenge oracle LR-Enc on two randomly chosen messages to obtain a challenge ciphertext. Then it queries both messages to the oracle Enc but on a public key which differs from the challenge public key only in the part affecting the auxiliary encryption function g. To determine the secret bit, the adversary simply compares the output of the core encryption function f for its queries. This is formalised in the following theorem.

Theorem 4.3.2. *For any PK-splittable public key encryption scheme* Pke*, there exists an adversary* $\mathcal{A}$ *such that*

$$\mathbf{Adv}^{\mathsf{lrINDCPRA}}(\mathcal{A}) = 1\,.$$

Proof. We construct the following adversary $\mathcal{A}$ playing game lrINDCPRA. Upon receiving the target public key $pk^* = (pk_g^*, pk_f^*)$, $\mathcal{A}$ chooses two messages m_0 and m_1 at random and queries (m_0, m_1) to LR-Enc to obtain a challenge ciphertext c. Subsequently, $\mathcal{A}$ runs KGen to obtain a public key $pk' = (pk'_g, pk'_f)$, sets $pk = (pk_g^*, pk'_f)$, and queries both (pk, m_0) and (pk, m_1) to Enc to obtain ciphertexts c_0 and c_1. Let c_0^f, c_1^f, and c^f be the core encryption function parts of the ciphertext c_0, c_1, and c, respectively. If $c^f = c_0^f$, $\mathcal{A}$ outputs 0. If $c^f = c_1^f$, it outputs 1.

Since Pke is a PK-splittable scheme, we have $c_0 = (f(pk_f^*, r^*, m_0), g(pk'_g, r^*))$ and $c_1 = (f(pk_f^*, r^*, m_1), g(pk'_g, r^*))$. The ciphertext c depends on the secret bit b of the game lrINDCPRA. If $b = 0$, c equals $(f(pk_f^*, r^*, m_0), g(pk_g^*, r^*))$ and if $b = 1$, it equals

$(f(pk_f^*, r^*, m_1), g(pk_g^*, r^*))$. Hence, if $b = 0$, the core encryption function part of the ciphertext c is equal to the core encryption function part of c_0. This yields

$$\Pr[\mathcal{A}^{\mathsf{lrINDCPRA}} \Rightarrow 0 \,|\, b = 0] = 1\,.$$

Likewise, if $b = 1$, the core encryption function parts of c and c_1 are equal, hence

$$\Pr[\mathcal{A}^{\mathsf{lrINDCPRA}} \Rightarrow 1 \,|\, b = 1] = 1 \iff \Pr[\mathcal{A}^{\mathsf{lrINDCPRA}} \Rightarrow 0 \,|\, b = 1] = 0\,.$$

This enables $\mathcal{A}$ to perfectly distinguish the cases $b = 0$ and $b = 1$, as

$$\begin{aligned}\mathbf{Adv}^{\mathsf{lrINDCPRA}}(\mathcal{A}) &= \left|\Pr[\mathcal{A}^{\mathsf{lrINDCPRA}} \Rightarrow 0 \,|\, b = 0] - \Pr[\mathcal{A}^{\mathsf{lrINDCPRA}} \Rightarrow 0 \,|\, b = 1]\right| \\ &= 1 - 0 \\ &= 1\,.\end{aligned}$$

It remains to argue that $\mathcal{A}$ is a valid adversary against lrINDCPRA. Since it queries m_0 and m_1 both to LR-Enc and Enc, it looks like $\mathcal{A}$ is not equality-pattern-respecting. However, the property equality-pattern-respecting only prohibits querying a message to LR-Enc which has been queried to Enc together with the target public key. Our adversary never queries Enc on the target public key since it replaces pk_g^*, i.e., the part that affects the auxiliary encryption function, for both queries. Hence, the set $\mathcal{E}$, from the definition of equality-pattern-respecting (cf. Definition 2.5.2), is empty which yields that $\mathcal{A}$ is an equality-pattern-respecting adversary. □

4.3.2 Real-World PKE Schemes that are LoR-IND-CPRA-Insecure

In this section we show several real-world schemes to be vulnerable against resetting attacks by showing that they are PK-splittable. We show this for the generic LWE-based PKE scheme; the code-based PKE schemes HQC, RQC, and ROLLO-II; and the standard hybrid encryption scheme.

Lattice-Based PKE Schemes

The canonical LWE-based PKE scheme is due to Regev [Reg05, Reg09]. A ciphertext consists of two LWE samples, once of which is added to the message. It is displayed in Figure 2.28.

It is easy to see that from the two ciphertext parts c_1 and c_2, only c_1 depends on the message. Furthermore, each entry of the public key (a and b) affects exactly one ciphertext part. Thus, this scheme is PK-splittable which is formalised in the lemma below.

Lemma 4.3.3. *The LWE-based PKE scheme (cf. Figure 2.28) is PK-splittable.*

Proof. Figure 4.14 describes the key generation and encryption algorithm of the canonical LWE-based PKE scheme as well as the scheme written as a PK-splittable scheme. □

Corollary 4.3.4. *Let* PKE *be the LWE-based public key encryption schemes. Then there exists an adversary $\mathcal{A}$ such that*

$$\mathbf{Adv}^{\mathsf{lrINDCPRA}}_{\mathrm{PKE}}(\mathcal{A}) = 1\,.$$

Proof. Follows directly from Theorem 4.3.2 and Lemma 4.3.3. □

$\underline{\texttt{KGen}(\lambda; r)}$

$\mathsf{a}, \mathsf{s}, \mathsf{e} \leftarrow r$
$\mathsf{b} \leftarrow \mathsf{as} + \mathsf{e}$
$pk \leftarrow (\mathsf{a}, \mathsf{b})$
$sk \leftarrow \mathsf{s}$
return (pk, sk)

$\underline{\texttt{Enc}(pk, m; r)}$

parse pk **as** (a, b)
$\mathsf{e}_1, \mathsf{e}_2, \mathsf{d} \leftarrow r$
$c_1 \leftarrow \mathsf{bd} + \mathsf{e}_1 + \mathsf{Encode}(m)$
$c_2 \leftarrow \mathsf{ad} + \mathsf{e}_2$
return $c \leftarrow (c_1, c_2)$

$\underline{\texttt{KGen}(\lambda; \overline{r})}$

$\mathsf{a}, \mathsf{s}, \mathsf{e} \leftarrow \overline{r}$
$\mathsf{b} \leftarrow \mathsf{as} + \mathsf{e}$
$pk_f \leftarrow \mathsf{b}$
$pk_g \leftarrow \mathsf{a}$
$sk \leftarrow \mathsf{s}$
return $((pk_g, pk_f), sk)$

$\underline{\texttt{Enc}(pk, m; r)}$

parse pk **as** (pk_g, pk_f)
$c_1 \leftarrow f(pk_f, r, m)$
$c_2 \leftarrow g(pk_g, r)$
return $c \leftarrow (c_1, c_2)$

$\underline{f(pk_f, r, m)}$

parse pk_f **as** b
$\mathsf{e}_1, \mathsf{e}_2, \mathsf{d} \leftarrow r$
return $\mathsf{bd} + \mathsf{e}_1 + \mathsf{Encode}(m)$

$\underline{g(pk_g, r)}$

parse pk_g **as** a
$\mathsf{e}_1, \mathsf{e}_2, \mathsf{d} \leftarrow r$
return $\mathsf{ad} + \mathsf{e}_2$

Figure 4.14: **Left:** LWE-based PKE scheme. **Right:** LWE-based PKE scheme written as a PK-splittable scheme with core encryption function f and auxiliary encryption function g.

Code-Based PKE Schemes

The code-based public key encryption schemes HQC [AAB+19a], RQC [AAB+19b], and ROLLO-II [ABD+19] were Round 2 candidates in the NIST post-quantum cryptography standardization process [NIST17].

For all schemes, only c_1 is affected by the message and the public key can be split into a core encryption function and auxiliary encryption function related part; for ROLLO-II there is no core encryption function related part of the public key. Note that the attack presented here is independent of the concrete sets from which vectors etc. are drawn. Due to this, we display the schemes in a simplified manner for which HQC and RQC are in fact identical.

Lemma 4.3.5. *The code-based public key encryption scheme HQC (cf. Figure 2.31) is PK-splittable.*

Proof. The key generation algorithm and encryption algorithm of HQC written as a PK-splittable scheme are displayed in Figure 4.15 which also recalls the base algorithms. □

Lemma 4.3.6. *The code-based public key encryption scheme RQC (cf. Figure 2.32) is PK-splittable.*

Proof. The key generation algorithm and encryption algorithm of RQC written as a PK-splittable scheme are displayed in Figure 4.16 which also recalls the base algorithms. □

Lemma 4.3.7. *The code-based public key encryption scheme ROLLO-II (cf. Figure 2.30) is PK-splittable.*

$\underline{\texttt{KGen}(\lambda;\overline{r})}$

$\mathbf{h},\mathbf{x},\mathbf{y},\mathbf{G} \leftarrow \overline{r}$
$\mathbf{s} \leftarrow \mathbf{x} + \mathbf{hy}$
$pk \leftarrow (\mathbf{h},\mathbf{s},\mathbf{G})$
$sk \leftarrow (\mathbf{x},\mathbf{y})$
return (pk, sk)

$\underline{\texttt{Enc}(pk, m; r)}$

parse pk **as** $(\mathbf{h},\mathbf{s},\mathbf{G})$
$\mathbf{r}_1,\mathbf{r}_2,\mathbf{e} \leftarrow r$
$c_1 \leftarrow m\mathbf{G} + \mathbf{sr}_2 + \mathbf{e}$
$c_2 \leftarrow \mathbf{r}_1 + \mathbf{hr}_2$
return $c \leftarrow (c_1, c_2)$

$\underline{\texttt{KGen}(\lambda;\overline{r})}$

$\mathbf{h},\mathbf{x},\mathbf{y},\mathbf{G} \leftarrow \overline{r}$
$\mathbf{s} \leftarrow \mathbf{x} + \mathbf{hy}$
$pk_f \leftarrow (\mathbf{s},\mathbf{G})$
$pk_g \leftarrow \mathbf{h}$
$sk \leftarrow (\mathbf{x},\mathbf{y})$
return $((pk_g, pk_f)sk)$

$\underline{\texttt{Enc}(pk, m; r)}$

parse pk **as** (pk_g, pk_f)
$c_1 \leftarrow f(pk_f, r, m)$
$c_2 \leftarrow g(pk_g, r)$
return $c \leftarrow (c_1, c_2)$

$\underline{f(pk_f, r, m)}$

parse pk_f **as** $(\mathbf{s},\mathbf{G})$
$\mathbf{r}_1,\mathbf{r}_2,\mathbf{e} \leftarrow r$
return $m\mathbf{G} + \mathbf{sr}_2 + \mathbf{e}$

$\underline{g(pk_g, r)}$

parse pk_g **as** $\mathbf{h}$
$\mathbf{r}_1,\mathbf{r}_2,\mathbf{e} \leftarrow r$
return $\mathbf{r}_1 + \mathbf{hr}_2$

Figure 4.15: **Left:** Code-based PKE scheme HQC. **Right:** HQC written as a PK-splittable scheme with core encryption function f and auxiliary encryption function g.

$\underline{\texttt{KGen}(\lambda;\overline{r})}$

$\mathbf{h},\mathbf{x},\mathbf{y},\mathbf{G} \leftarrow \overline{r}$
$\mathbf{s} \leftarrow \mathbf{x} + \mathbf{hy}$
$pk \leftarrow (\mathbf{h},\mathbf{s},\mathbf{G})$
$sk \leftarrow (\mathbf{x},\mathbf{y})$
return (pk, sk)

$\underline{\texttt{Enc}(pk, m; r)}$

parse pk **as** $(\mathbf{h},\mathbf{s},\mathbf{G})$
$\mathbf{r}_1,\mathbf{r}_2,\mathbf{e} \leftarrow r$
$c_1 \leftarrow m\mathbf{G} + \mathbf{sr}_2 + \mathbf{e}$
$c_2 \leftarrow \mathbf{r}_1 + \mathbf{hr}_2$
return $c \leftarrow (c_1, c_2)$

$\underline{\texttt{KGen}(\lambda;\overline{r})}$

$\mathbf{h},\mathbf{x},\mathbf{y},\mathbf{G} \leftarrow \overline{r}$
$\mathbf{s} \leftarrow \mathbf{x} + \mathbf{hy}$
$pk_f \leftarrow (\mathbf{s},\mathbf{G})$
$pk_g \leftarrow \mathbf{h}$
$sk \leftarrow (\mathbf{x},\mathbf{y})$
return $((pk_g, pk_f)sk)$

$\underline{\texttt{Enc}(pk, m; r)}$

parse pk **as** (pk_g, pk_f)
$c_1 \leftarrow f(pk_f, r, m)$
$c_2 \leftarrow g(pk_g, r)$
return $c \leftarrow (c_1, c_2)$

$\underline{f(pk_f, r, m)}$

parse pk_f **as** $(\mathbf{s},\mathbf{G})$
$\mathbf{r}_1,\mathbf{r}_2,\mathbf{e} \leftarrow r$
return $m\mathbf{G} + \mathbf{sr}_2 + \mathbf{e}$

$\underline{g(pk_g, r)}$

parse pk_g **as** $\mathbf{h}$
$\mathbf{r}_1,\mathbf{r}_2,\mathbf{e} \leftarrow r$
return $\mathbf{r}_1 + \mathbf{hr}_2$

Figure 4.16: **Left:** Code-based PKE scheme RQC. **Right:** RQC written as a PK-splittable scheme with core encryption function f and auxiliary encryption function g.

Proof. Figure 4.17 recalls the algorithms $\texttt{KGen}$ and $\texttt{Enc}$ of ROLLO-II and further displays them written as a PK-splittable scheme. □

$\underline{\texttt{KGen}(\lambda; r)}$

$\mathbf{x}, \mathbf{y} \leftarrow r$
$\mathbf{h} \leftarrow \mathbf{x}^{-1}\mathbf{y} \bmod \mathsf{P}$
$sk \leftarrow (\mathbf{x}, \mathbf{y})$
$pk \leftarrow \mathbf{h}$
return (pk, sk)

$\underline{\texttt{Enc}(pk, m; r)}$

$\mathbf{e}_1, \mathbf{e}_2 \leftarrow r$
$\mathsf{E} \leftarrow \mathsf{Supp}(\mathbf{e}_1, \mathbf{e}_2)$
$c_1 \leftarrow m \oplus \mathsf{H}(\mathsf{E})$
$c_2 \leftarrow \mathbf{e}_1 + \mathbf{e}_2\mathbf{h} \bmod \mathsf{P}$
return $c \leftarrow (c_1, c_2)$

$\underline{\texttt{KGen}(\lambda; \overline{r})}$

$\mathbf{x}, \mathbf{y} \leftarrow \overline{r}$
$\mathbf{h} \leftarrow \mathbf{x}^{-1}\mathbf{y}$
$pk_g \leftarrow \mathbf{h}$
$pk_f \leftarrow \varepsilon$
$sk \leftarrow (\mathbf{x}, \mathbf{y})$
return $((pk_g, pk_f), sk)$

$\underline{\texttt{Enc}(pk, m; r)}$

parse pk **as** (pk_g, pk_f)
$c_1 \leftarrow f(pk_f, r, m)$
$c_2 \leftarrow g(pk_g, r)$
return $c \leftarrow (c_1, c_2)$

$\underline{f(pk_f, r, m)}$

parse pk_f **as** $\emptyset$
$\mathbf{e}_1, \mathbf{e}_2 \leftarrow r$
$\mathsf{E} \leftarrow \mathsf{Supp}(\mathbf{e}_1, \mathbf{e}_2)$
return $m \oplus \mathsf{H}(\mathsf{E})$

$\underline{g(pk_g, r)}$

parse pk_g **as** $\mathbf{h}$
$\mathbf{e}_1, \mathbf{e}_2 \leftarrow r$
return $\mathbf{e}_1 + \mathbf{e}_2\mathbf{h}$

Figure 4.17: **Left:** Code-based PKE scheme ROLLO-II. **Right:** ROLLO-II written as a PK-splittable scheme with core encryption function f and auxiliary encryption function g.

Corollary 4.3.8. *Let* $\textsc{Pke} \in \{HQC, RQC, ROLLO\text{-}II\}$. *There exists an adversary* $\mathcal{A}$ *such that*

$$\mathbf{Adv}^{\mathsf{IrINDCPRA}}_{\textsc{Pke}}(\mathcal{A}) = 1\,.$$

Proof. Follows directly from combining Theorem 4.3.2 with Lemma 4.3.5, Lemma 4.3.6, and Lemma 4.3.7 for HQC, RQC, and ROLLO-II, respectively. □

Hybrid Encryption

Now we turn our attention towards the security of the hybrid encryption scheme against resetting attacks. As discussed above, the attack proposed in [Yil10b] exploits the insecurity of the one-time pad when a key is used more than once. The attack no longer works when using an arbitrary symmetric key encryption scheme instead of the one-time pad, as the adversary does not learn the symmetric key from a single query. However, we show that the hybrid encryption scheme (cf. Figure 2.33) is PK-splittable, irrespective of the underlying schemes. This shows that any instantiation of it is susceptible to resetting attacks.

Lemma 4.3.9. *Let* $\Sigma^P = (\texttt{KGen}^P, \texttt{Enc}^P, \texttt{Dec}^P)$ *be a public key encryption scheme and* $\Sigma^S = (\texttt{Enc}^S, \texttt{Dec}^S)$ *be a symmetric key encryption scheme. The resulting hybrid encryption scheme* $\textsc{Pke} = (\texttt{KGen}, \texttt{Enc}, \texttt{Dec})$, *see Figure 2.33, is a PK-splittable scheme.*

Proof. The scheme written as a PK-splittable scheme is displayed in Figure 4.18. The core encryption function corresponds to the underlying symmetric key encryption scheme which is used to encrypt the actual message. The auxiliary encryption function g corresponds to the underlying public key encryption scheme used to encrypt the symmetric key. □

$\underline{\mathtt{KGen}(\lambda)}$

$(pk, sk) \leftarrow \mathtt{KGen}^P(\lambda)$
return (pk, sk)

$\underline{\mathtt{Enc}(pk, m; r)}$

$k, r^*, r' \leftarrow r$
$c_1 \leftarrow \mathtt{Enc}^S(k, m; r')$
$c_2 \leftarrow \mathtt{Enc}^P(pk, k; r^*)$
return $c \leftarrow (c_1, c_2)$

$\underline{\mathtt{KGen}(\lambda; \overline{r})}$

$(pk, sk) \leftarrow \mathtt{KGen}^P(\lambda; \overline{r})$
$pk_g \leftarrow pk$
$pk_f \leftarrow \varepsilon$
return $((pk_g, pk_f), sk)$

$\underline{\mathtt{Enc}(pk, m; r)}$

parse pk **as** (pk_g, pk_f)
$c_1 \leftarrow f(pk_f, r, m)$
$c_2 \leftarrow g(pk_g, r)$
return $c \leftarrow (c_1, c_2)$

$\underline{f(pk_f, r, m)}$

parse pk_f **as** $\emptyset$
$k, r^*, r \leftarrow r$
return $\mathtt{Enc}^S(k, m; r')$

$\underline{g(pk_g, r)}$

parse pk_g **as** pk
$k, r^*, r \leftarrow r$
return $\mathtt{Enc}^P(pk, k; r^*)$

Figure 4.18: **Left:** Hybrid encryption scheme combining a public key encryption scheme $(\mathtt{KGen}^P, \mathtt{Enc}^P, \mathtt{Dec}^P)$ and a symmetric key encryption scheme $(\mathtt{Enc}^S, \mathtt{Dec}^S)$. **Right:** Hybrid encryption scheme written as a PK-splittable scheme with core encryption function f and auxiliary encryption function g.

Corollary 4.3.10. *Let* PKE *be the hybrid encryption scheme instantiated with an arbitrary public key encryption scheme and symmetric key encryption scheme. Then, there exists an adversary $\mathcal{A}$ such that*

$$\mathbf{Adv}^{\mathsf{lrINDCPRA}}_{\mathrm{PKE}}(\mathcal{A}) = 1\,.$$

Proof. Follows directly from Theorem 4.3.2 and Lemma 4.3.9. □

4.3.3 Achieving LoR-IND-CPRA Security

Along with the introduction of the LoR-IND-CPRA security notion, Yilek also provides a generic transformation to achieve LoR-IND-CPRA security. The transformation requires a pseudorandom function $\mathcal{F}$. A scheme is transformed by not using r as random coins for encryption, instead, the random coins are derived as $\mathcal{F}_r(pk, m)$. That is, the pseudorandom function is keyed by the actual random coins and evaluated on the public key and the message to encrypt. Provided that $\mathcal{F}$ is a pseudorandom function, Yilek shows that this is sufficient to achieve LoR-IND-CPRA security.

4.4 Security Notions against Resetting Attacks

In this section, we show that security against adversaries making a single query to the challenge oracle implies security against adversaries making multiple queries to the challenge

oracle. This confirms the claim made in [Yil10b] by using a different proof that does not suffer from the issue pointed out in [PSS14] and also mentioned in [Sib15]. In Section 4.4.1 we recall the proof given in [Yil10b] and its flaw that has been identified in [PSS14]. We construct an adversary which distinguishes the hybrid games almost perfectly, which entails that the existing proof cannot be fixed. We then give a different proof for the claim in Section 4.4.2. Furthermore, we define security against resetting attacks in a real-or-random sense in Section 4.4.3 and show that the straightforward hybrid argument allows simplification to a single challenge. In Section 4.4.4, we show that the left-or-right and real-or-random notions for resetting attacks are equivalent. As a side effect, these results yield a modular proof for the main result in this section, which we give in Section 4.4.5.

4.4.1 Shortcomings of Yilek's Proof

We specify two special queries, which are not forbidden by Definition 2.5.2. It turns out, that these queries are the ones that invalidate the proof in [Yil10b]. First, after making a query (m_0, m_1) to $\mathsf{LR\text{-}Enc}$, the adversary can make the same query to $\mathsf{LR\text{-}Enc}$. We call this a repeating query. Second, after making a query (m_0, m_1) to $\mathsf{LR\text{-}Enc}$, the adversary can query (m_1, m_0) to $\mathsf{LR\text{-}Enc}$. We call this a flipping query.

The proof in [Yil10b] uses a sequence of hybrid games $\mathsf{H}_0, \ldots, \mathsf{H}_q$ (cf. Figure 4.19). In H_i, the first i queries are answered by encrypting the right message m_1, while the remaining $q - i$ queries are answered by encrypting the left message m_0. By construction, H_0 and H_q equal game $\mathsf{lrINDCPRA}$ with secret bit $b = 0$ and $b = 1$, respectively. To bound two consecutive hybrids H_{i-1} and H_i, the following reduction $\mathcal{R}_\mathsf{i}$ is constructed. Each query by $\mathcal{A}$ to Enc is forwarded by $\mathcal{R}_\mathsf{i}$ to its own oracle Enc. The first $i-1$ challenge queries are answered by querying the left message to Enc, the last $q-i$ challenge queries by querying the right message to Enc, in both cases together with the target public key pk^*. The i^{th} challenge query is forwarded by $\mathcal{R}_\mathsf{i}$ to its own challenge oracle $\mathsf{LR\text{-}Enc}$.

We now elaborate why the reduction does not work if the adversary makes a repeating query or a flipping query.[22] Let (m_0, m_1) be the i^{th} challenge query by $\mathcal{A}$ that is forwarded by the reduction to its own challenge oracle. Let $j, k > i$ and, without loss of generality, assume that the j^{th} and k^{th} query are (m_0, m_1) and (m_1, m_0), respectively. Thus, the j^{th} query is a repeating query and the k^{th} query is a flipping query. For the j^{th} query, the reduction would query its oracle Enc on m_0 and for the k^{th} query it would query it on m_1. Neither of these two queries is allowed, as both m_0 and m_1 have been queried to $\mathsf{LR\text{-}Enc}$. Thus, this makes the reduction not equality-pattern-respecting.

At the first glance, this looks like an issue in the reduction, but we show that the issue lies in the hybrid games. More precisely, we give an equality-pattern-respecting adversary that can distinguish two consecutive hybrid games with probability 1. This adversary rules out any proof using these hybrid games, thereby preventing a simple fix of the proof in [Yil10b].

Lemma 4.4.1. *Let* $\textsc{Pke} = (\mathtt{KGen}, \mathtt{Enc}, \mathtt{Dec})$ *be a perfectly correct public key encryption scheme and* H_i *be the hybrid game displayed in Figure 4.19. For any* $i \in [q]$*, there exists*

[22] The issue described in [PSS14] corresponds to the issue for flipping queries we show here.

an adversary $\mathcal{A}_{\mathsf{i}}$ *such that*

$$\left|\Pr[\mathcal{A}_{\mathsf{i}}^{\mathsf{H}_{i-1}} \Rightarrow 0] - \Pr[\mathcal{A}_{\mathsf{i}}^{\mathsf{H}_i} \Rightarrow 0]\right| = 1\,.$$

Proof. For $i \in [q-1]$, we construct the following adversary $\mathcal{A}_{\mathsf{i}}$. The first $i-1$ and the last $q-i-1$ queries are randomly chosen messages that have never been queried. For the i^{th} query, $\mathcal{A}_{\mathsf{i}}$ picks two messages m_0 and m_1 at random and queries $\mathsf{LR\text{-}Enc}$ on (m_0, m_1) to obtain a ciphertext c_i. For the $(i+1)^{\text{th}}$ query, $\mathcal{A}_{\mathsf{i}}$ invokes $\mathsf{LR\text{-}Enc}$ on the flipping query (m_1, m_0), resulting in a ciphertext c_{i+1}. If $c_i = c_{i+1}$, $\mathcal{A}_{\mathsf{i}}$ outputs 1. Otherwise, i.e., $c_i \neq c_{i+1}$, it outputs 0.

In game H_{i-1}, both the i^{th} and the $(i+1)^{\text{th}}$ query are answered by encrypting the left message. Since the i^{th} and $(i+1)^{\text{th}}$ queries by $\mathcal{A}_{\mathsf{i}}$ are (m_0, m_1) and (m_1, m_0), this yields $c_i \leftarrow \mathtt{Enc}(pk^*, m_0; r^*)$ and $c_{i+1} \leftarrow \mathtt{Enc}(pk^*, m_1; r^*)$. Then we have $c_i \neq c_{i+1}$ since the scheme is perfectly correct and hence

$$\Pr[\mathcal{A}_{\mathsf{i}}^{\mathsf{H}_{i-1}} \Rightarrow 0] = 1\,.$$

In game H_i, the i^{th} query is answered by encrypting the right message instead while the $(i+1)^{\text{th}}$ query is still answered by encrypting the left message. This yields $c_i \leftarrow \mathtt{Enc}(pk^*, m_1; r^*)$ and $c_{i+1} \leftarrow \mathtt{Enc}(pk^*, m_1; r^*)$. Hence we have $c_i = c_{i+1}$ which yields

$$\Pr[\mathcal{A}_{\mathsf{i}}^{\mathsf{H}_i} \Rightarrow 0] = 0\,.$$

Adversary $\mathcal{A}_{\mathsf{q}}$ performs $q-2$ challenge queries on random messages. Then it picks two fresh messages m_0 and m_1—meaning they have not been queried before—followed by querying first (m_1, m_0) and then (m_0, m_1) resulting in ciphertexts c_{q-1} and c_q, respectively. If $c_{q-1} = c_q$, $\mathcal{A}_{\mathsf{q}}$ outputs 0, otherwise, i.e., $c_i \neq c_{i+1}$, $\mathcal{A}_{\mathsf{q}}$ outputs 1.

The same argument as above yields $c_{q-1} \leftarrow \mathtt{Enc}(pk^*, m_0; r^*)$ and $c_q \leftarrow \mathtt{Enc}(pk^*, m_0; r^*)$ in H_{q-1}. Hence $c_{q-1} = c_q$ which leads to

$$\Pr[\mathcal{A}_{\mathsf{q}}^{\mathsf{H}_{q-1}} \Rightarrow 0] = 1\,.$$

Likewise, for H_q it holds that $c_{q-1} \leftarrow \mathtt{Enc}(pk^*, m_0; r^*)$ and $c_q \leftarrow \mathtt{Enc}(pk^*, m_1; r^*)$. Due to PKE being perfectly correct it follows that $c_{q-1} \neq c_q$ and thus

$$\Pr[\mathcal{A}_{\mathsf{q}}^{\mathsf{H}_q} \Rightarrow 0] = 0\,.$$

Hence, for any $i \in [q]$ we have

$$\left|\Pr[\mathcal{A}_{\mathsf{i}}^{\mathsf{H}_{i-1}} \Rightarrow 0] - \Pr[\mathcal{A}_{\mathsf{i}}^{\mathsf{H}_i} \Rightarrow 0]\right| = 1 - 0 = 1\,.$$

This concludes the proof. □

When considering public key encryption schemes with negligible probability for decryption failures, the distinguishing advantage decreases negligibly. That is because the ciphertexts c_i and c_{i+1} for the messages m_0 and m_1 might be equal in H_{i-1} (or in H_q when considering $\mathcal{A}_{\mathsf{q}}$). Nevertheless, two consecutive hybrids can be distinguished almost perfectly.

Game H_i	**oracle** $\mathsf{LR\text{-}Enc}(m_0, m_1)$	**oracle** $\mathsf{Enc}(pk, m)$
$b \leftarrow\!\!\$\ \{0,1\}$	$c \leftarrow c + 1$	$c \leftarrow \mathtt{Enc}(pk, m; r^*)$
$c \leftarrow 0$	**if** $c \leq i$	**return** c
$(pk^*, sk^*) \leftarrow\!\!\$\ \mathtt{KGen}()$	$\quad c \leftarrow \mathtt{Enc}(pk^*, m_1; r^*)$	
$r^* \leftarrow\!\!\$\ \mathcal{R}$	**else**	
$b' \leftarrow \mathcal{A}^{\mathsf{LR\text{-}Enc},\mathsf{Enc}}(pk^*)$	$\quad c \leftarrow \mathtt{Enc}(pk^*, m_0; r^*)$	
	return c	

Figure 4.19: Hybrid games in the proof in [Yil10b].

4.4.2 Alternative Proof for Yilek's Claim

Having established that the proof approach in [Yil10b] does not work, we now turn our attention towards providing a different proof for the statement. Recall that the flawed proof uses a single hybrid argument over the number of queries by the adversary. To deal with the issue of flipping and repeating queries, we change the overall approach as follows. First, instead of a single hybrid argument where we switch from encryption of the left messages to encryption of the right messages, we use two hybrid arguments: one where we first switch from encrypting the left messages to encrypting random messages and one where we switch from encrypting random messages to encrypting the right messages. Second, instead of doing the hybrid argument over the number of queries, we do the hybrid argument over the number of distinct queries, i.e., non-repeating queries, by the adversary. The former change avoids the issue of flipping queries while the latter change circumvents the issue of repeating queries.

Below we state our main result. It shows that, in the case of resetting attacks, security against adversaries making a single query to their challenge oracle implies security against adversaries making multiple queries to their challenge oracle. It confirms the claim made in [Yil10b] at the cost of an additional factor of 2 in the bound.

Theorem 4.4.2. *Let* $\textsc{Pke} = (\mathtt{KGen}, \mathtt{Enc}, \mathtt{Dec})$ *be a public key encryption scheme and the game* $\mathsf{lrINDCPRA}$ *be defined as in Figure 2.24. Then for any equality-pattern-respecting adversary* $\mathcal{A}$ *making* q *distinct queries to* $\mathsf{LR\text{-}Enc}$*, there exists an equality-pattern-respecting adversary* $\mathcal{R}$ *making* 1 *query to* $\mathsf{LR\text{-}Enc}$*, such that*

$$\mathbf{Adv}^{\mathsf{lrINDCPRA}}_{\textsc{Pke}}(\mathcal{A}) \leq 2q\, \mathbf{Adv}^{\mathsf{lrINDCPRA}}_{\textsc{Pke}}(\mathcal{R})\,.$$

Proof. We prove the theorem using hybrid games $\mathsf{L}_0, \ldots, \mathsf{L}_q, \mathsf{R}_0, \ldots, \mathsf{R}_q$, which are displayed in Figure 4.20. In game L_i, the first i distinct challenge queries are answered by encrypting a random message, while the remaining $q - i$ distinct challenge queries are answered by encrypting the left message. Game R_i is defined analogously except that the right message, rather than the left message, is encrypted. Note that, in any game, repeating queries are answered by looking up the previous response in the set $\mathcal{Q}$ (cf. Figure 4.20). From this description we can deduce that hybrid games L_0 and R_0 correspond to the game $\mathsf{lrINDCPRA}$ with secret bit $b = 0$ and $b = 1$, respectively. Furthermore, hybrid games L_q and R_q are identical, as they both answer all q distinct challenge queries by encrypting a

random message. Thus we have

$$\begin{aligned}
\mathbf{Adv}^{\mathsf{lrINDCPRA}}(\mathcal{A}) &= \Big|\Pr[\mathcal{A}^{\mathsf{lrINDCPRA}} \Rightarrow 0 \,|\, b = 0] - \Pr[\mathcal{A}^{\mathsf{lrINDCPRA}} \Rightarrow 0 \,|\, b = 1]\Big| \\
&= \Big|\Pr[\mathcal{A}^{\mathsf{L}_0} \Rightarrow 0] - \Pr[\mathcal{A}^{\mathsf{R}_0} \Rightarrow 0]\Big| \\
&\leq \Big|\Pr[\mathcal{A}^{\mathsf{L}_0} \Rightarrow 0] - \Pr[\mathcal{A}^{\mathsf{L}_q} \Rightarrow 0]\Big| + \Big|\Pr[\mathcal{A}^{\mathsf{R}_q} \Rightarrow 0] - \Pr[\mathcal{A}^{\mathsf{R}_0} \Rightarrow 0]\Big| \\
&\leq \sum_{i=1}^{q} \Big|\Pr[\mathcal{A}^{\mathsf{L}_{i-1}} \Rightarrow 0] - \Pr[\mathcal{A}^{\mathsf{L}_i} \Rightarrow 0]\Big| \\
&\quad + \sum_{i=1}^{q} \Big|\Pr[\mathcal{A}^{\mathsf{R}_i} \Rightarrow 0] - \Pr[\mathcal{A}^{\mathsf{R}_{i-1}} \Rightarrow 0]\Big| \,.
\end{aligned}$$

To bound the consecutive hybrids L_{i-1} and L_i for $i \in [q]$, we construct the following adversary $\mathcal{B}_\mathsf{i}$ playing lrINDCPRA. On input pk^*, $\mathcal{B}_\mathsf{i}$ runs $\mathcal{A}$ on input pk^*. When $\mathcal{A}$ makes a query m to Enc, $\mathcal{B}_\mathsf{i}$ forwards m to its own oracle Enc and the response back to $\mathcal{A}$. For the first $i-1$ distinct queries $(m_0^1, m_1^1), \ldots, (m_0^{i-1}, m_1^{i-1})$ by $\mathcal{A}$ to LR-Enc, $\mathcal{B}_\mathsf{i}$ responds by querying its oracle Enc on a random message $m_* \leftarrow\!\!\$\, \mathcal{M}$ and the target public key pk^* to obtain a ciphertext that it forwards to $\mathcal{A}$. For the i^{th} distinct query (m_0^i, m_1^i), $\mathcal{B}_\mathsf{i}$ chooses $m_* \leftarrow\!\!\$\, \mathcal{M}$, queries (m_0^i, m_*) to its oracle LR-Enc, and sends the response back to $\mathcal{A}$. For the last $q-i$ distinct queries $(m_0^{i+1}, m_1^{i+1}), \ldots, (m_0^q, m_1^q)$ by $\mathcal{A}$ to LR-Enc, $\mathcal{B}_\mathsf{i}$ queries its oracle Enc $q-i$ times on $m_0^{i+1}, \ldots, m_0^q$ together with the target public key pk^* and forwards the response to $\mathcal{A}$. For all these distinct queries, $\mathcal{B}_\mathsf{i}$ stores the queried messages along with the returned ciphertext in a set $\mathcal{Q}$. For any repeating query, $\mathcal{B}_\mathsf{i}$ returns the same ciphertext which it looks up in the set $\mathcal{Q}$. When $\mathcal{A}$ halts and outputs a bit b', $\mathcal{B}_\mathsf{i}$ outputs b'.

Recall that the proof in [Yil10b] does not hold since the reduction has to make a forbidden query. Since our reduction makes only one query to LR-Enc, we have to ensure that it never queries its oracle Enc on one of the messages queried to LR-Enc. The critical part is the simulation of the oracle LR-Enc for $\mathcal{A}$ using the oracle Enc. By construction, $\mathcal{B}_\mathsf{i}$ queries LR-Enc on (m_0^i, m_*), where m_0^i is from the i^{th} query (m_0^i, m_1^i) to LR-Enc by $\mathcal{A}$ and m_* is a randomly chosen message by $\mathcal{B}_\mathsf{i}$. Prior to this query, $\mathcal{B}_\mathsf{i}$ invokes Enc only on random messages, hence the probability that one of these is equal to either m_0^i or m_* is negligible. Subsequent to its challenge query, $\mathcal{B}_\mathsf{i}$ queries Enc on the left messages that $\mathcal{A}$ queries to its challenge oracle. We know that each query of the form $(m_0^i, \cdot)$ is a repeating query, i.e., (m_0^i, m_1^i). Any query (m_0^i, m') where $m' \neq m_1^i$ is excluded as $\mathcal{A}$ is equality-pattern-respecting. Since repeating queries are answered using the set $\mathcal{Q}$, they do not involve an oracle query by $\mathcal{B}_\mathsf{i}$. Hence $\mathcal{B}_\mathsf{i}$ is a valid adversary, i.e., equality-pattern-respecting, playing lrINDCPRA.

By construction we have that $\mathcal{B}_\mathsf{i}$ simulates game L_{i-1} and game L_i for $\mathcal{A}$ if its own challenge bit b is 0 and 1, respectively. This yields

$$\begin{aligned}
\Big|\Pr[\mathcal{A}^{\mathsf{L}_{i-1}} \Rightarrow 0] - \Pr[\mathcal{A}^{\mathsf{L}_i} \Rightarrow 0]\Big| &= \Big|\Pr[\mathcal{B}_\mathsf{i}^{\mathsf{lrINDCPRA}} \Rightarrow 0 \,|\, b = 0] - \Pr[\mathcal{B}_\mathsf{i}^{\mathsf{lrINDCPRA}} \Rightarrow 0 \,|\, b = 1]\Big| \\
&= \mathbf{Adv}^{\mathsf{lrINDCPRA}}(\mathcal{B}_\mathsf{i}) \,.
\end{aligned}$$

Analogously, we can construct adversaries $\mathcal{C}_\mathsf{i}$ to bound consecutive hybrid games R_{i-1} and R_i with the following two differences. First, for the i^{th} distinct query (m_0^i, m_1^i) by $\mathcal{A}$, $\mathcal{C}_\mathsf{i}$ queries its own challenge oracle $\mathsf{LR\text{-}Enc}$ on (m_*, m_1^i), for a randomly chosen message m_*, and sends the result back to $\mathcal{A}$. For the last $q-i$ distinct queries $(m_0^{i+1}, m_1^{i+1}), \ldots, (m_0^q, m_1^q)$ by $\mathcal{A}$, $\mathcal{C}_\mathsf{i}$ invokes its oracle Enc on the right messages, i.e., $m_1^{i+1}, \ldots, m_1^q$, and sends the responses back to $\mathcal{A}$. At the end, $\mathcal{C}_\mathsf{i}$ outputs b', where b' is the output of $\mathcal{A}$.

Just as above, $\mathcal{C}_\mathsf{i}$ simulates game R_i if $b = 0$ and game R_{i-1} if $b = 1$, which yields

$$\begin{aligned}\Big|\Pr[\mathcal{A}^{\mathsf{R}_i} \Rightarrow 0] - \Pr[\mathcal{A}^{\mathsf{R}_{i-1}} \Rightarrow 0]\Big| &= \Big|\Pr[\mathcal{C}_\mathsf{i}^{\mathsf{lrINDCPRA}} \Rightarrow 0 \,|\, b = 0] - \Pr[\mathcal{C}_\mathsf{i}^{\mathsf{lrINDCPRA}} \Rightarrow 0 \,|\, b = 1]\Big| \\ &= \mathbf{Adv}^{\mathsf{lrINDCPRA}}(\mathcal{C}_\mathsf{i})\,.\end{aligned}$$

We define $\mathcal{B}$ and $\mathcal{C}$ to be the adversary that picks $i \leftarrow\!\!\$\, [q]$ and then behaves as $\mathcal{B}_\mathsf{i}$ and $\mathcal{C}_\mathsf{i}$, respectively. By a standard hybrid argument [FM21, MF21], we get

$$\begin{aligned}\mathbf{Adv}^{\mathsf{lrINDCPRA}}(\mathcal{A}) &\le \sum_{i=1}^{q} \Big|\Pr[\mathcal{A}^{\mathsf{L}_{i-1}} \Rightarrow 0] - \Pr[\mathcal{A}^{\mathsf{L}_i} \Rightarrow 0]\Big| \\ &\quad + \sum_{i=1}^{q} \Big|\Pr[\mathcal{A}^{\mathsf{R}_i} \Rightarrow 0] - \Pr[\mathcal{A}^{\mathsf{R}_{i-1}} \Rightarrow 0]\Big| \\ &= \sum_{i=1}^{q} \mathbf{Adv}^{\mathsf{lrINDCPRA}}(\mathcal{B}_\mathsf{i}) + \sum_{i=1}^{q} \mathbf{Adv}^{\mathsf{lrINDCPRA}}(\mathcal{C}_\mathsf{i}) \\ &\le q\,\mathbf{Adv}^{\mathsf{lrINDCPRA}}(\mathcal{B}) + q\,\mathbf{Adv}^{\mathsf{lrINDCPRA}}(\mathcal{C})\,.\end{aligned}$$

Defining $\mathcal{R}$ as the adversary with the higher advantage among $\mathcal{B}$ and $\mathcal{C}$ finally yields

$$\begin{aligned}\mathbf{Adv}^{\mathsf{lrINDCPRA}}(\mathcal{A}) &\le q\,\mathbf{Adv}^{\mathsf{lrINDCPRA}}(\mathcal{B}) + q\,\mathbf{Adv}^{\mathsf{lrINDCPRA}}(\mathcal{C}) \\ &\le 2q\,\mathbf{Adv}^{\mathsf{lrINDCPRA}}(\mathcal{R})\,.\end{aligned}$$

This proves the claim. □

We briefly discuss why the issue of the proof in [Yil10b] does not occur here. In [Yil10b], the reduction did not work as the adversary can query $\mathsf{LR\text{-}Enc}$ on (m_0, m_1) in the i^{th} query and later make a flipping query (m_1, m_0). Then the reduction would query (m_0, m_1) to its oracle $\mathsf{LR\text{-}Enc}$ and later m_1 to its oracle Enc, which makes the reduction not equality-pattern-respecting. The adversary can do the same in our case. However, our reduction invokes its oracle $\mathsf{LR\text{-}Enc}$ on (m_0, m_*), rather than (m_0, m_1), which allows it to later invoke its oracle Enc on m_1. The issue of repeating queries does not occur as we do the hybrid arguments over the number of distinct queries which leads to a reduction which never makes an oracle query when the adversary makes a repeating query.

4.4.3 Real-or-Random Security against Resetting Attacks

In Figure 4.21 we give the real-or-random security game $\mathsf{rrINDCPRA}$ against resetting attacks. It is easy to see that this notion is unachievable, even when imposing the standard equality-pattern-restrictions on the adversary (cf. Definition 2.5.2). An adversary simply

Games $\mathsf{L}_i, \mathsf{R}_i$

$b \leftarrow\!\!\$\ \{0,1\}$
$c \leftarrow 0$
$\mathcal{Q} \leftarrow \emptyset$
$(pk^*, sk^*) \leftarrow\!\!\$\ \mathtt{KGen}()$
$r^* \leftarrow\!\!\$\ \mathcal{R}$
$b' \leftarrow \mathcal{A}^{\mathsf{LR\text{-}Enc},\mathsf{Enc}}(pk^*)$

oracle $\mathsf{Enc}(pk, m)$

$c \leftarrow \mathtt{Enc}(pk, m; r^*)$
return c

oracle $\mathsf{LR\text{-}Enc}(m_0, m_1)$ in L_i

if $\exists c$ **s.t.** $(m_0, m_1, c) \in \mathcal{Q}$
  return c
$c \leftarrow c + 1$
if $c \leq i$
  $m_* \leftarrow\!\!\$\ \mathcal{M}$
  $c \leftarrow \mathtt{Enc}(pk^*, m_*; r^*)$
else
  $c \leftarrow \mathtt{Enc}(pk^*, m_0; r^*)$
$\mathcal{Q} \leftarrow\!\!\cup\ (m_0, m_1, c)$
return c

oracle $\mathsf{LR\text{-}Enc}(m_0, m_1)$ in R_i

if $\exists c$ **s.t.** $(m_0, m_1, c) \in \mathcal{Q}$
  return c
$c \leftarrow c + 1$
if $c \leq i$
  $m_* \leftarrow\!\!\$\ \mathcal{M}$
  $c \leftarrow \mathtt{Enc}(pk^*, m_*; r^*)$
else
  $c \leftarrow \mathtt{Enc}(pk^*, m_1; r^*)$
$\mathcal{Q} \leftarrow\!\!\cup\ (m_0, m_1, c)$
return c

Figure 4.20: Hybrid games L_i and R_i used to prove Theorem 4.4.2.

queries the same message twice to its real-or-random oracle RR-Enc. In case $b = 0$, it obtains the same ciphertext since the same message is encrypted under the same randomness. In case $b = 1$, however, the ciphertexts will be different (even though they used the same randomness) as two different messages will be encrypted. This allows the adversary to distinguish with overwhelming probability.

Game rrINDCPRA

$b \leftarrow\!\!\$\ \{0,1\}$
$(pk^*, sk^*) \leftarrow\!\!\$\ \mathtt{KGen}()$
$r^* \leftarrow\!\!\$\ \mathcal{R}$
$b' \leftarrow \mathcal{A}^{\mathsf{RR\text{-}Enc},\mathsf{Enc}}(pk^*)$
return $(b' = b)$

oracle $\mathsf{RR\text{-}Enc}(m)$

if $b = 0$
  $m_* \leftarrow\!\!\$\ \mathcal{M}$
  $c \leftarrow \mathtt{Enc}(pk^*, m_*; r^*)$
else
  $c \leftarrow \mathtt{Enc}(pk^*, m; r^*)$
return c

oracle $\mathsf{Enc}(pk, m)$

$c \leftarrow \mathtt{Enc}(pk, m; r^*)$
return c

Figure 4.21: Security game to define RoR-IND-CPRA security.

To circumvent this trivial win, we can use the security game as displayed in Figure 4.22. The game keeps a list of messages queried to the real-or-random oracle and ensures that, in case $b = 1$, the same random message is encrypted when the adversary queries the same challenge message. In the game displayed in Figure 4.22 this is done via the table f. This prevents the trivial attack described above. However, it also renders repeating queries obsolete as it does not give the adversary any additional information.

Due to this, we stick with the security game displayed in Figure 4.21 and exclude repeating queries via the definition of equality-pattern-respecting. Instead of having two different definitions for equality-pattern-respecting, depending on the left-or-right and real-or-random case, we use the unified definition below. It extends Definition 2.5.2 to also cover real-or-random adversaries.

Game rrINDCPRA	**oracle** RR-Enc(m)	**oracle** Enc(pk, m)
$b \leftarrow_\$ \{0,1\}$	**if** $b = 0$	$c \leftarrow \texttt{Enc}(pk, m; r^*)$
$(pk^*, sk^*) \leftarrow_\$ \texttt{KGen}()$	**if** $f[m] = \bot$	**return** c
$r^* \leftarrow_\$ \mathcal{R}$	$f[m] \leftarrow_\$ \mathcal{M}$	
$b' \leftarrow \mathcal{A}^{\mathsf{RR\text{-}Enc},\mathsf{Enc}}(pk^*)$	$m_* \leftarrow f[m]$	
return $(b' = b)$	$c \leftarrow \texttt{Enc}(pk^*, m_*; r^*)$	
	else	
	$c \leftarrow \texttt{Enc}(pk^*, m; r^*)$	
	return c	

Figure 4.22: Real-or-random security dealing with repeating queries.

Definition 4.4.3. *Let $\mathcal{A}_{\mathsf{LR}}$ and $\mathcal{A}_{\mathsf{RR}}$ be adversaries playing game* lrINDCPRA *and game* rrINDCPRA*, respectively. Let further $\mathcal{E}_{LR}$ (resp. $\mathcal{E}_{RR}$) be the set of messages m such that $\mathcal{A}_{\mathsf{LR}}$ (resp. $\mathcal{A}_{\mathsf{RR}}$) makes a query (pk^*, m) to* Enc*. Let $(m_0^1, m_1^1), \ldots, (m_0^q, m_1^q)$ be the queries that $\mathcal{A}_{\mathsf{LR}}$ makes to* LR-Enc *and $m^1, \ldots, m^q$ be the queries of $\mathcal{A}_{\mathsf{RR}}$ to* RR-Enc*.*
We say that $\mathcal{A}_{\mathsf{LR}}$ is equality-pattern-respecting *if*

- *for all $b \in \{0,1\}$ and $i \in [q]$, $m_b^i \notin \mathcal{E}_{LR}$ and*
- *for all $b \in \{0,1\}$ and $i \neq j$, $m_b^i = m_b^j \implies m_{1-b}^i = m_{1-b}^j$.*

We say that $\mathcal{A}_{\mathsf{RR}}$ is equality-pattern-respecting *if*

- *for all $i \in [q]$, $m^i \notin \mathcal{E}_{RR}$ and*
- *for $i \neq j$ it holds that $m^i \neq m^j$.*

Just as for the left-or-right case, the real-or-random (RoR-IND-CPRA) advantage of an adversary is defined as its advantage over random guessing scaled to the interval from 0 to 1.

Definition 4.4.4. *Let* PKE $= (\texttt{KGen}, \texttt{Enc}, \texttt{Dec})$ *be a public key encryption scheme and the game* rrINDCPRA *be defined as in Figure 4.21. For any equality-pattern-respecting adversary $\mathcal{A}$, its* rrINDCPRA *advantage is defined as*

$$\mathbf{Adv}^{\mathsf{rrINDCPRA}}(\mathcal{A}) \coloneqq \left|2 \Pr[\mathsf{rrINDCPRA}^{\mathcal{A}} \Rightarrow 1] - 1\right| .$$

We now show that real-or-random security against adversaries making a single query to the challenge oracle RR-Enc implies real-or-random security against adversaries making multiple queries to the challenge oracle RR-Enc. It turns out that the standard hybrid technique, which failed in the left-or-right case, works for this setting. This stems from the fact that the real-or-random adversary submits only one message to its challenge oracle. This makes it impossible to make a flipping query, which was the main issue in the left-or-right setting.

Theorem 4.4.5. *Let* PKE *be a public key encryption scheme and the game* rrINDCPRA *be defined as in Figure 4.21. Then, for any equality-pattern-respecting (cf. Definition 4.4.3) adversary $\mathcal{A}$ making q queries to its challenge oracle* RR-Enc*, there exists an equality-pattern-respecting adversary $\mathcal{B}$ making one query to* RR-Enc *such that*

$$\mathbf{Adv}^{\mathsf{rrINDCPRA}}_{\mathrm{PKE}}(\mathcal{A}) \leq q\,\mathbf{Adv}^{\mathsf{rrINDCPRA}}_{\mathrm{PKE}}(\mathcal{B})\,.$$

Proof. The theorem is proven using a standard hybrid argument. We use $q+1$ hybrid games $\mathsf{H}_0, \ldots, \mathsf{H}_q$ which are displayed in Figure 4.23. In hybrid H_i, the first i queries are answered by encrypting a random message, while the remaining $q-i$ queries are answered by encrypting the message provided by the adversary. It holds that

$$\begin{aligned}
\mathbf{Adv}^{\mathsf{rrINDCPRA}}(\mathcal{A}) &= \left|2\,\Pr[\mathsf{rrINDCPRA}^{\mathcal{A}} \Rightarrow 1] - 1\right| \\
&= \Big|\Pr[\mathcal{A}^{\mathsf{rrINDCPRA}} \Rightarrow 1 \mid b = 1] - \Pr[\mathcal{A}^{\mathsf{rrINDCPRA}} \Rightarrow 1 \mid b = 0]\Big| \\
&= \Big|\Pr[\mathcal{A}^{\mathsf{H}_0} \Rightarrow 1] - \Pr[\mathcal{A}^{\mathsf{H}_q} \Rightarrow 1]\Big| \\
&\leq \sum_{i=1}^{q} \Big|\Pr[\mathcal{A}^{\mathsf{H}_{i-1}} \Rightarrow 1] - \Pr[\mathcal{A}^{\mathsf{H}_i} \Rightarrow 1]\Big|\,.
\end{aligned}$$

We construct the following adversary $\mathcal{B}_{\mathsf{i}}$ to bound the distinguishing advantage between H_{i-1} and H_i for $i \in [q]$. It runs $\mathcal{A}$ on the same public key pk^* it receives as input and answers any query to Enc by $\mathcal{A}$ using its own oracle Enc. For the first $i-1$ challenge queries $m^1, \ldots, m^{i-1}$ by $\mathcal{A}$, $\mathcal{B}_{\mathsf{i}}$ invokes its oracle Enc on randomly chosen messages. For the i^{th} query m^i, $\mathcal{B}_{\mathsf{i}}$ invokes its own challenge oracle on m^i and sends the obtained ciphertext back to $\mathcal{A}$. For the last $q-i$ queries $m^{i+1}, \ldots, m^q$ by $\mathcal{A}$, $\mathcal{B}_{\mathsf{i}}$ invokes its oracle Enc on $m^{i+1}, \ldots, m^q$.

The adversary $\mathcal{B}_{\mathsf{i}}$ perfectly simulates H_{i-1} and H_i for $\mathcal{A}$ if its own challenge bit is 1 and 0, respectively. To show that $\mathcal{B}_{\mathsf{i}}$ is equality-pattern-respecting, we have to show that it never repeats a query to RR-Enc and never queries a message to both RR-Enc and Enc. The former is trivial as $\mathcal{B}_{\mathsf{i}}$ makes exactly one query to RR-Enc. For the latter, recall that only challenge queries by $\mathcal{A}$ that are answered using Enc can be problematic as otherwise, $\mathcal{A}$ would not be equality-pattern-respecting. Since $\mathcal{A}$ is equality-pattern-respecting, all its queries are on fresh messages, which yields that $\mathcal{B}_{\mathsf{i}}$ never queries its challenge message also to Enc. Hence we have

$$\begin{aligned}
\mathbf{Adv}^{\mathsf{rrINDCPRA}}(\mathcal{A}) &\leq \sum_{i=1}^{q} \Big|\Pr[\mathcal{A}^{\mathsf{H}_{i-1}} \Rightarrow 1] - \Pr[\mathcal{A}^{\mathsf{H}_i} \Rightarrow 1]\Big| \\
&= \sum_{i=1}^{q} \mathbf{Adv}^{\mathsf{rrINDCPRA}}(\mathcal{B}_{\mathsf{i}})\,.
\end{aligned}$$

By a standard hybrid argument [FM21, MF21], we define $\mathcal{B}$ to be the adversary that picks $i \leftarrow\!\!\$\ [q]$ and then behaves as $\mathcal{B}_{\mathsf{i}}$ which yields

$$\mathbf{Adv}^{\mathsf{rrINDCPRA}}(\mathcal{A}) \leq \sum_{i=1}^{q} \mathbf{Adv}^{\mathsf{rrINDCPRA}}(\mathcal{B}_{\mathsf{i}}) \leq q\,\mathbf{Adv}^{\mathsf{rrINDCPRA}}(\mathcal{B})\,.$$

This proves the claim. □

Game H_i	**oracle** LR-Enc(m)	**oracle** Enc(pk, m)
$b \leftarrow\!\!\$\ \{0,1\}$	$c \leftarrow c + 1$	$c \leftarrow \mathtt{Enc}(pk, m; r^*)$
$c \leftarrow 0$	**if** $c \leq i$	**return** c
$(pk^*, sk^*) \leftarrow\!\!\$\ \mathtt{KGen}()$	$\quad m_* \leftarrow\!\!\$\ \mathcal{M}$	
$r^* \leftarrow\!\!\$\ \mathcal{R}$	$\quad c \leftarrow \mathtt{Enc}(pk^*, m_*; r^*)$	
$b' \leftarrow \mathcal{A}^{\mathsf{LR\text{-}Enc},\mathsf{Enc}}(pk^*)$	**else**	
	$\quad c \leftarrow \mathtt{Enc}(pk^*, m; r^*)$	
	return c	

Figure 4.23: Hybrid games used to prove Theorem 4.4.5.

4.4.4 Equivalence between Real-or-Random and Left-or-Right Security

In this section we show that our real-or-random security notion against resetting attacks is equivalent to the left-or-right security notion given in [Yil10b]. We show the equivalence by proving two lemmas. The former shows that left-or-right security implies real-or-random security, while the latter shows that real-or-random security implies left-or-right security. For both, we use the unified definition of equality-pattern-respecting (cf. Definition 4.4.3).

Lemma 4.4.6. *Let* PKE *be a public key encryption scheme and the games* rrINDCPRA *and* lrINDCPRA *be defined as in Figure 4.21 and Figure 2.24, respectively. Then, for any equality-pattern-respecting adversary $\mathcal{A}$ making q queries to its challenge oracle* RR-Enc*, there exists an equality-pattern-respecting adversary $\mathcal{B}$ making q distinct queries to* LR-Enc *such that*

$$\mathbf{Adv}^{\mathsf{rrINDCPRA}}_{\mathrm{PKE}}(\mathcal{A}) \leq \mathbf{Adv}^{\mathsf{lrINDCPRA}}_{\mathrm{PKE}}(\mathcal{B})\,.$$

Proof. The adversary $\mathcal{B}$ runs $\mathcal{A}$ on the same public key pk^* that it receives. The oracle Enc for $\mathcal{A}$ is simulated by $\mathcal{B}$ using its own oracle Enc. When $\mathcal{A}$ makes a challenge query m_i, $\mathcal{B}$ chooses a random message m_* and invokes its own challenge oracle of (m_*, m_i) and sends the response back to $\mathcal{A}$. When $\mathcal{A}$ outputs its guess b', $\mathcal{B}$ outputs b' as its own guess.

It is easy to see that $\mathcal{B}$ perfectly simulates game rrINDCPRA with secret bit b for $\mathcal{A}$, where b coincides with the secret bit that $\mathcal{B}$ is asked to find in game lrINDCPRA. Likewise, $\mathcal{B}$ is equality-pattern-respecting. Every challenge query is on two fresh messages since one is the message by $\mathcal{A}$ which never repeats a challenge message and the other one is always sampled at random, i.e., $\mathcal{B}$ makes only distinct queries. Every query to Enc stems from a query to Enc by $\mathcal{A}$. □

Lemma 4.4.7. *Let* PKE *be a public key encryption scheme and the games* rrINDCPRA *and* lrINDCPRA *be defined as in Figure 4.21 and Figure 2.24, respectively. Then, for any equality-pattern-respecting adversary $\mathcal{A}$ making q distinct queries to its challenge oracle* LR-Enc*, there exists an equality-pattern-respecting adversary $\mathcal{B}$ making q queries to* RR-Enc *such that*

$$\mathbf{Adv}^{\mathsf{lrINDCPRA}}_{\mathrm{PKE}}(\mathcal{A}) \leq 2\,\mathbf{Adv}^{\mathsf{rrINDCPRA}}_{\mathrm{PKE}}(\mathcal{B})\,.$$

Proof. This proof resembles the proof of Theorem 4.4.2. We use three games G_0, G_1, and G_2 which are displayed in Figure 4.24. It holds that G_0 corresponds to game $\mathsf{lrINDCPRA}$ with secret bit $b = 0$. The same holds for G_2 except that the secret bit b is 1. This yields

$$\begin{aligned}
\mathbf{Adv}^{\mathsf{lrINDCPRA}}(\mathcal{A}) &= \left|2\Pr[\mathsf{lrINDCPRA}^{\mathcal{A}} \Rightarrow 1] - 1\right| \\
&= \left|\Pr[\mathcal{A}^{\mathsf{lrINDCPRA}} \Rightarrow 0 \,|\, b = 0] - \Pr[\mathcal{A}^{\mathsf{lrINDCPRA}} \Rightarrow 0 \,|\, b = 1]\right| \\
&= \left|\Pr[\mathcal{A}^{\mathsf{G}_0} \Rightarrow 0] - \Pr[\mathcal{A}^{\mathsf{G}_2} \Rightarrow 0]\right| \\
&\leq \left|\Pr[\mathcal{A}^{\mathsf{G}_0} \Rightarrow 0] - \Pr[\mathcal{A}^{\mathsf{G}_1} \Rightarrow 0]\right| + \left|\Pr[\mathcal{A}^{\mathsf{G}_1} \Rightarrow 0] - \Pr[\mathcal{A}^{\mathsf{G}_2} \Rightarrow 0]\right| .
\end{aligned}$$

To bound the first difference, we construct the following adversary $\mathcal{B}_1$. It runs $\mathcal{A}$ on the same public key pk^*. Queries by $\mathcal{A}$ to Enc are forwarded by $\mathcal{B}_1$ to its own oracle Enc as are the responses back to $\mathcal{A}$. For every distinct challenge query (m_0^i, m_1^i) by $\mathcal{A}$, $\mathcal{B}_1$ invokes its own challenge oracle $\mathsf{RR\text{-}Enc}$ on m_0^i and sends the received ciphertext back to $\mathcal{A}$. Every repeating query is answered with the same ciphertext using a set $\mathcal{Q}$. When $\mathcal{A}$ outputs a bit b', $\mathcal{B}_1$ forwards $1 - b'$ as its own output.

If the secret bit b in game RoR-IND-CPRA equals 1, $\mathcal{B}_1$ perfectly simulates G_0 for $\mathcal{A}$ as it receives back the encryption of the left message. On the other hand, if $b = 0$, $\mathcal{B}_1$ receives back the encryption of a random message, hence it perfectly simulates G_1. It remains to argue that $\mathcal{B}_1$ is equality-pattern-respecting. It clearly does not query Enc on any message that it queries to $\mathsf{RR\text{-}Enc}$ as this would entail that $\mathcal{A}$ is not equality-pattern-respecting. It also never repeats a query to $\mathsf{RR\text{-}Enc}$ since it queries it exactly on the left messages that $\mathcal{A}$ queries to $\mathsf{LR\text{-}Enc}$. Since repeating queries by $\mathcal{A}$ are answered using set $\mathcal{Q}$, the only possibility would be that $\mathcal{A}$ makes two queries (m_0^i, m_1^i) and (m_0^j, m_1^j) with $m_0^i = m_0^j$ and $m_1^i \neq m_1^j$. As an equality-pattern-respecting adversary, $\mathcal{A}$ never makes such queries. Thus, we have

$$\begin{aligned}
\left|\Pr[\mathcal{A}^{\mathsf{G}_0} \Rightarrow 0] - \Pr[\mathcal{A}^{\mathsf{G}_1} \Rightarrow 0]\right| &= \left|\Pr[\mathcal{B}_{\mathsf{i}}^{\mathsf{rrINDCPRA}} \Rightarrow 1 \,|\, b = 1] - \Pr[\mathcal{B}_{\mathsf{i}}^{\mathsf{rrINDCPRA}} \Rightarrow 1 \,|\, b = 0]\right| \\
&= \mathbf{Adv}^{\mathsf{rrINDCPRA}}(\mathcal{B}_1) .
\end{aligned}$$

In the same way, we can construct an adversary $\mathcal{B}_2$ to bound the advantage of $\mathcal{A}$ in distinguishing G_1 and G_2. The difference is that $\mathcal{B}_2$ forwards the right message of $\mathcal{A}$ as its own challenge message to $\mathsf{RR\text{-}Enc}$ and when $\mathcal{A}$ outputs a bit b', $\mathcal{B}_2$ outputs b' as well. It holds that $\mathcal{B}_2$ perfectly simulates G_1 and G_2 if its own challenge bit b equals 0 and 1, respectively. Equality-pattern-respecting follows by the same argument as above. This yields

$$\begin{aligned}
\left|\Pr[\mathcal{A}^{\mathsf{G}_1} \Rightarrow 0] - \Pr[\mathcal{A}^{\mathsf{G}_2} \Rightarrow 0]\right| &= \left|\Pr[\mathcal{B}_{\mathsf{i}}^{\mathsf{rrINDCPRA}} \Rightarrow 0 \,|\, b = 0] - \Pr[\mathcal{B}_{\mathsf{i}}^{\mathsf{rrINDCPRA}} \Rightarrow 0 \,|\, b = 1]\right| \\
&= \mathbf{Adv}^{\mathsf{rrINDCPRA}}(\mathcal{B}_2) .
\end{aligned}$$

Let $\mathcal{B}$ be the adversary with higher advantage among $\mathcal{B}_1$ and $\mathcal{B}_2$, then we have

$$\begin{aligned}
\mathbf{Adv}^{\mathsf{lrINDCPRA}}(\mathcal{A}) &\leq \left|\Pr[\mathcal{A}^{\mathsf{G}_0} \Rightarrow 0] - \Pr[\mathcal{A}^{\mathsf{G}_1} \Rightarrow 0]\right| + \left|\Pr[\mathcal{A}^{\mathsf{G}_1} \Rightarrow 0] - \Pr[\mathcal{A}^{\mathsf{G}_2} \Rightarrow 0]\right| \\
&= \mathbf{Adv}^{\mathsf{rrINDCPRA}}(\mathcal{B}_1) + \mathbf{Adv}^{\mathsf{rrINDCPRA}}(\mathcal{B}_2) \\
&\leq 2\,\mathbf{Adv}^{\mathsf{rrINDCPRA}}(\mathcal{B}) .
\end{aligned}$$

This proves the claim. □

Game G_i

$b \leftarrow\!\!\$\ \{0,1\}$
$\mathcal{Q} \leftarrow \emptyset$
$(pk^*, sk^*) \leftarrow\!\!\$\ \mathtt{KGen}()$
$r^* \leftarrow\!\!\$\ \mathcal{R}$
$b' \leftarrow \mathcal{A}^{\mathsf{LR\text{-}Enc},\mathsf{Enc}}(pk^*)$

oracle $\mathsf{LR\text{-}Enc}(m_0, m_1)$ in G_0

if $\exists c$ **s.t.** $(m_0, m_1, c) \in \mathcal{Q}$
  return c
$c \leftarrow \mathtt{Enc}(pk, m_0; r^*)$
$\mathcal{Q} \leftarrow\!\!\cup\ (m_0, m_1, c)$
return c

oracle $\mathsf{LR\text{-}Enc}(m_0, m_1)$ in G_1

if $\exists c$ **s.t.** $(m_0, m_1, c) \in \mathcal{Q}$
  return c
$m_* \leftarrow\!\!\$\ \mathcal{M}$
$c \leftarrow \mathtt{Enc}(pk, m_*; r^*)$
$\mathcal{Q} \leftarrow\!\!\cup\ (m_0, m_1, c)$
return c

oracle $\mathsf{LR\text{-}Enc}(m_0, m_1)$ in G_2

if $\exists c$ **s.t.** $(m_0, m_1, c) \in \mathcal{Q}$
  return c
$c \leftarrow \mathtt{Enc}(pk, m_1; r^*)$
return c

oracle $\mathsf{Enc}(pk, m)$

$c \leftarrow \mathtt{Enc}(pk, m; r^*)$
return c

Figure 4.24: Games used to prove Lemma 4.4.7.

The proof is very much akin the one for Theorem 4.4.2, where G_0 and G_2 correspond to L_0 and R_0, respectively, while G_1 equals $\mathsf{L}_q = \mathsf{R}_q$.

4.4.5 A Modular Proof for Yilek's Claim

Having established the equivalence between the left-or-right and the real-or-random cases via Lemma 4.4.6 and Lemma 4.4.7, we can leverage Theorem 4.4.5 to prove Theorem 4.4.2 more modularly.

Alternative proof of Theorem 4.4.2. From Lemma 4.4.7, Theorem 4.4.5, and Lemma 4.4.6 there exist equality-pattern-respecting adversaries $\mathcal{B}$, $\mathcal{C}$, and $\mathcal{R}$, respectively, where

- $\mathcal{B}$ makes q real-or-random queries,
- $\mathcal{C}$ makes 1 real-or-random query, and
- $\mathcal{R}$ makes 1 left-or-right query,

such that

$$\begin{aligned}
\mathbf{Adv}^{\mathsf{lrINDCPRA}}_{\mathrm{PKE}}(\mathcal{A}) &\leq 2\,\mathbf{Adv}^{\mathsf{rrINDCPRA}}_{\mathrm{PKE}}(\mathcal{B}) && \text{(by Lemma 4.4.7)}\\
&\leq 2q\,\mathbf{Adv}^{\mathsf{rrINDCPRA}}_{\mathrm{PKE}}(\mathcal{C}) && \text{(by Theorem 4.4.5)}\\
&\leq 2q\,\mathbf{Adv}^{\mathsf{lrINDCPRA}}_{\mathrm{PKE}}(\mathcal{R})\,. && \text{(by Lemma 4.4.6)}
\end{aligned}$$

This proves the claim. □

4.5 Summary and Outlook

In this chapter we developed two RKA security notions for AEAD schemes and analysed three generic constructions (N1, N2, and N3) with respect to the new notion. We showed an inherent limitation due to the separate keys and simultaneously showed that security boils down to the underlying encryption scheme and message authentication code. The analysis of the $\textsc{Se}[\mathcal{F},\mathcal{G}]$ and $\textsc{Mac}[\mathcal{H},\mathcal{F}']$ construction further revealed that RKA secure AEAD schemes can be constructed from RKA secure pseudorandom functions. Hence, constructing pseudorandom functions secure against a large class of RKD functions is a worthwhile target.

For sake of efficiency, modern AEAD schemes, for instance finalists in the CAESAR competition [DEMS16, Wu16, WP16, JNPS16], are based on direct constructions rather than the generic N-schemes. An open question is whether these schemes achieve RKA security, in particular, with respect to which set of RKD functions.

We further showed that the folklore simplification to a single left-or-right challenge also holds in case of resetting attacks. This confirms the claim in [Yil10b], for which the proof was identified as flawed [PSS14], at the cost of an additional factor 2 compared to the original claim. Security notions against the more general type of related-randomness attacks are explicitly formalised to allow multiple challenge queries [PSS14, MS18]. This effectively stems from the identification of the flawed proof for resetting attacks. We conjecture that our proof approach for resetting attacks also applies to related-randomness attacks which would allow for the same simplification to a single challenge query.

Finally, we defined a class of public key encryption schemes which are vulnerable to resetting attacks—hence also vulnerable to related-randomness attacks. We demonstrate this class to be quite general by showing that many PKE schemes are covered by it. The LoR-IND-CPRA security notion is strong in that it allows the adversary to submit arbitrary public keys to its encryption oracle. This allows, in particular, to query public keys for which there is no corresponding secret key. One can consider a weaker variant of the notion which excludes such malformed keys as done in [PSS14] for RRA. An open question is whether some of the schemes achieve security with respect to this weaker variant.

While PKE schemes can be fixed to achieve LoR-IND-CPRA, e.g., using the PRF transformation [Yil10b], we are not aware of any PKE scheme that achieves LoR-IND-CPRA without any changes. An open question is whether such schemes exist at all.

Part II

Security against Quantum Attacks

CHAPTER 5

POST-QUANTUM SECURITY

In this chapter, we develop a lifting theorem for post-quantum security of public key encryption schemes using random oracles. We further show that the generic FGHF′ *construction and the sponge-based instantiation* SLAE *developed in Chapter 3 achieve post-quantum security. Finally, we provide a post-quantum security proof for a generic construction of deterministic wallets and show that Yao's garbled circuits are post-quantum secure if the underlying building blocks are. The lifting theorem is based on [KS20a] while the analysis of* SLAE *is based on [JS22]. The results for the deterministic wallet construction and Yao's garbled circuits are based on [ADE+20] and [BDK+20], respectively.*

Contents

5.1 Lifting Theorem for IND-CPA Security 156
5.2 Post-Quantum Security of SLAE 163
5.3 Post-Quantum Security of Deterministic Wallets 171
5.4 Post-Quantum Security of Yao's Garbled Circuits 179
5.5 Summary and Outlook 186

The development of large-scale quantum computers will have significant impact on cryptography. Shor [Sho94] developed an efficient quantum algorithm for factoring and solving discrete logarithms in 1994. Two years later, in 1996, Grover [Gro96] developed a quantum search algorithm that allows for a quadratic speed-up, compared to classical algorithms.

The effect of Shor's algorithm on public key cryptography is tremendous as it completely breaks currently used public key cryptography such as RSA [RSA78], ElGamal [ElG85] and ECDSA [KD13]. This threat sparked the research field of post-quantum cryptography [BBD09]: cryptography that is usable on classical computers while withstanding

attackers with quantum computing power. In particular, adversaries are limited to local quantum computation. At the moment, there are five main types of post-quantum cryptography:

1. lattice-based cryptography [Ajt96, Reg05, Reg09],
2. code-based cryptography [McE78, Nie86],
3. hash-based cryptography [Lam79, Mer90],
4. multivariate cryptography [MI88, Pat95, Pat96, KPG99], and
5. isogeny-based cryptography [Cou06, RS06b, JDF11].

While some of the areas were developed even before Shor's algorithm, they received significantly less research attention than, for instance, RSA [RSA78]. In the last two decades, however, attention towards these areas increased tremendously, as they are believed to be candidate constructions that can withstand attackers with quantum computing power. To deal with the glooming threat of Shor's algorithm, the National Institute of Standards and Technology (NIST) initiated a standardization process for post-quantum cryptographic primitives [NIST17].

In contrast to Shor's algorithm, Grover's algorithm does not break currently used cryptography. But it threatens symmetric primitives by improving brute-force attacks on symmetric keys. Also finding collisions for hash functions can be improved using Grover's algorithm. To maintain security against quantum attackers the key length of symmetric primitives as well as the output length of hash functions are typically doubled [CJL+16].

Following the provable security paradigm, simple switching to, say, lattice-based primitives for public key cryptography and doubling the key length for symmetric primitives is not sufficient. To be confident in the security, proofs need to consider adversaries with quantum computing power. For many schemes this turns out to be easy as the classical proof works in the same way even for quantum adversaries. This mainly follows from the fact that the adversary is restricted to local quantum computing power; any communication with the (classical) challenger can only be classical.[23] But there are two exceptions to this.

The first exception are security proofs which rely on techniques that do not apply to quantum adversaries. The prominent example for this is rewinding. Here, the reduction first runs the adversary to obtain an output, then rewinds it to a certain point, and runs the adversary again to obtain another output. From the two outputs, the reduction can then solve some hardness assumption. Rewinding is commonly used for zero-knowledge proofs [GMR89] and appears for instance in lattice-based signatures [Lyu09, Lyu12] when applying the Fiat-Shamir transformation [FS87] and, in turn, the forking lemma [PS00, BN06]. Due to the no-cloning theorem [WZ82, Die82], a reduction cannot simply copy the state of the adversary and then run it again from this state. Furthermore, inverting the adversary up to the point of rewinding is challenging due to the destructive nature of measurements. While arbitrary rewinding is impossible, several works provide some positive results for rewinding quantum adversaries [Wat09, Unr12, ARU14].

[23] This is in sharp contrast to quantum security that we consider in Chapter 6.

The second exception are security proofs for which the security notion changes by granting a quantum adversary additional power. The prominent example for this are random oracles, where a quantum adversary gets superposition access while a classical adversary gets only classical access. To facilitate a security proof, primitives like hash functions are often idealised as random oracles. In that case, a proof is conducted in the random oracle model (ROM) [BR93]. Since random oracles idealise primitives that the adversary can evaluate locally a quantum adversary can compute it in superposition, recall that hash functions are public. To reflect this in security proofs, Boneh et al. [BDF+11] introduced the quantum random oracle model (QROM) which grants superposition access to parties with quantum computing power. Even though the QROM has become the defacto standard for post-quantum primitives that rely on random oracles, often schemes are accompanied only by a classical proof in the ROM. This includes a variety of primitives [LN17, EEE20, BLO18, TSS+18, TKS+19, LAZ19, HKLN20, SSZ17]. The same applies to sponge constructions but, unlike public key primitives based on post-quantum hardness assumptions, they are typically not claimed to be post-quantum secure. A crucial question is whether security in the ROM implies security in the QROM. Unfortunately, Yamakawa and Zhandry showed that arbitrary lifting from the ROM to the QROM is impossible [YZ21]. Their result is related to a proof of quantumness [BCM+18] and their separation is based on the LWE-based construction given in [BKVV20]. Therefore, it is important to study the post-quantum security of the aforementioned primitives in the QROM rather than the ROM, either by suitable lifting results—as done in [YZ21]—or by direct proofs in the QROM—as done for the signature scheme qTESLA [ABB+17].

In this chapter we focus on the case of random oracles; the case of rewinding does not occur for any of the proofs described here. In the following we further describe some of the challenges that come with the QROM.

A security proof in the ROM ignores the quantum computing power of an adversary; therefore it is insufficient to ensure post-quantum security, even for schemes based on quantum hard assumptions. Instead, the proof needs to be conducted in the QROM. However, several techniques from the ROM do not work in the QROM but, fortunately, alternative techniques were developed. In the ROM, a reduction can lazy sample a random oracle, i.e., rather than fixing a random oracle a priori, the reduction samples only output values that the adversary queries, hence the name "lazy". In the QROM, a single superposition query allows the adversary to query the whole domain of a random oracle. This thwarts the lazy sampling approach as the reduction would need to sample an exponential number of outputs. Zhandry [Zha12b] showed that a quantum adversary making q (superposition) queries cannot distinguish between a quantum random oracle and one simulated using a $2q$-wise independent function. Another example is query extraction. In the ROM, a reduction learns the values that the adversary queries and can simply copy them. In the QROM, the no-cloning theorem thwarts copying queries of the adversary. In cases, where the reduction needs to extract a specific input that the adversary queries, Unruh's one-way to hiding (O2H) lemma [Unr15c] provides a tool to do so. The seminal work by Zhandry [Zha19a] provides a technique to record superposition queries.

The above argument—that is, not considering the quantum computing power of the adversary—also applies to symmetric primitives; in particular for sponge-based primitives where the underlying transformation (or permutation) is idealised. Examples are several

of the finalists in the NIST standardization process for lightweight cryptography [NIST15]: ASCON [DEMS20], Elephant [BCDM20], ISAP [DEM+20a], PHOTON-Beetle [BCD+20], and SPARKLE [BBdS+20].[24] So far, none of these schemes is analysed with respect to quantum adversaries. A first step towards this goal was very recently done by Alagic et al. [ABKM21], who showed post-quantum security of the Even-Mansour cipher [EM97], which underpins, for instance, Elephant [BCDM20]. Jaeger et al. [JST21] showed post-quantum security of the FX construction [KR01], however, only for non-adaptive adversaries.

Contribution

In this chapter, we provide post-quantum security proofs for several primitives.

We develop a lifting theorem for public key encryption schemes that use random oracles. We identify a class of encryption schemes and provide requirements under which a proof considering classical adversaries in the ROM also holds against quantum adversaries in the QROM. We use these results to prove the code-based public key encryption scheme ROLLO-II [ABD+19] and the lattice-based public key encryption scheme LARA [Ban19] post-quantum secure.

We further show that the sponge-based AEAD scheme Slae developed in Chapter 3 is post-quantum secure. More precisely, we show that the post-quantum security of Slae boils down to the post-quantum security of the underlying sponge-based primitives SlFunc, SPrg, and SvHash. We then show that these primitives are secure against adversaries having superposition access to the underlying transformation ρ.

Finally, we look into more complex constructions rather than individual primitives by looking at constructions used in blockchain technologies and multi-party computation, respectively. First, we analyse the generic construction for deterministic wallets from signature schemes with rerandomizable keys proposed by Das et al. [DFL19]. The construction achieves post-quantum security if the underlying signature scheme does, but the proof requires some changes to also hold against quantum adversaries in the QROM. Then, we scrutinise the post-quantum security of Yao's garbled circuits (GC), the famous secure two-party computation (STPC) protocol. We show that the post-quantum security of the protocol follows from the classical proof by Lindell and Pinkas [LP09]. As for the underlying encryption scheme of the protocol, we adapt the proof sketch from [LP09] to the QROM, which shows that the scheme can be instantiated with post-quantum symmetric key encryption schemes, such as SlEnc.

Related Work

Since the introduction of the quantum random oracle model [BDF+11], there has been a lot of progress towards restoring techniques from the ROM to the QROM or developing alternative techniques. Initially, the QROM [BDF+11] required quantum secure pseudorandom functions to simulate a quantum random oracle which was shown to exist by Zhandry [Zha12a]. Zhandry also showed that this assumption is not necessary if a

[24] We note that TinyJAMBU [WH20] is an exception as it relies on a keyed permutation for the sponge construction, hence access to this permutation is still classical.

bound for the number of queries is known a priori, by showing that a $2q$-wise independent function is perfectly indistinguishable from a quantum random oracle when q queries are allowed [Zha12b]. The introduction of semi-constant distributions [Zha12b] allows to inject a challenge into a subset of a random oracle domain such that it (1) is sufficiently small that the adversary cannot detect it and (2) large enough that the adversary has a significant chance of obtaining the injected challenge. Zhandry [Zha12a] further showed, via so-called small-range distributions, how good an adversary can distinguish a quantum random oracle from an oracle that is drawn from such a small-range distribution. Finally, Unruh developed the one-way to hiding (O2H) lemma which provides a bound on the distinguishing advantage of an adversary between two quantum random oracles which differ at a certain input. Subsequent variants of the O2H lemma provide better bounds at the cost of more restricted applicability [Unr15c, Unr14, Unr15b, AHU19, BHH+19, KSS+20]. In [GHHM21], the authors provide results on adaptively reprogramming a quantum random oracle.

Along with the QROM, Boneh et al. [BDF+11] define history-free reductions. These are specific types of reductions in the ROM which they show to also hold in the QROM. However, many signature schemes, in particular those following the Fiat-Shamir transformation do not come with history-free reductions. There is a line of research regarding the post-quantum security of Fiat-Shamir signatures [DFG13, KLS18, LZ19, DFMS19, Unr17, DFM20, Cha19]. Hallgren et al. [HSS11] provide an abstraction called simple hybrid argument for which they show that classical proofs also work for quantum adversaries. Later, Song [Son14] provided a general framework for lifting security proofs that also covers the aforementioned simple hybrid argument. However, to apply the framework to schemes using random oracles, one needs to come up with a proof that transforms an adversary in the QROM into an adversary in the ROM which is generally the critical step. Zhang et al. [ZYF+19] lift security for committed-programming reductions from the ROM to the QROM. Recently, and subsequent to the lifting theorem developed in this chapter, Yamakawa and Zhandry [YZ21] provide a lifting theorem for signature schemes which is based entirely on the scheme and independent of the classical proof.

Research in the realm of post-quantum security of sponges mainly targets the security of hash functions. The first result addresses sponge-based hash functions based on random transformations or non-invertible random permutations [CBH+18]. The ultimate goal is a post-quantum proof for SHA-3 [FIP15] which is targeted both by Unruh [Unr21][25] and Czajkowski [Cza21] using Zhandry's compressed oracle technique [Zha19a].

Roadmap

In Section 5.1 we develop the lifting theorem for public key encryption schemes. Post-quantum security of Slae is covered in Section 5.2. Section 5.3 and Section 5.4 deal with the post-quantum security of the deterministic wallet construction and Yao's garbled circuits, respectively. We conclude with a summary and further research directions in Section 5.5.

[25] Note that the current version of the paper (20211202:133649) contains an erratum describing a mistake in one of the proofs which breaks most of the results.

5.1 Lifting Theorem for IND-CPA Security

In this section we develop a lifting theorem for the IND-CPA security of public key encryption schemes that use random oracles. Precisely, we develop requirements for schemes and the classical proof in the ROM such that a proof also holds in the QROM. For simplicity, we consider security against adversaries making a single challenge query. In this case, the classical and post-quantum variant of the security games can be written as displayed in Figure 5.1. We consider a two-stage adversary $\mathcal{A}$ consisting of a message generator $\mathcal{A}_{\mathsf{m}}$ and a distinguisher $\mathcal{A}_{\mathsf{d}}$. The former generates and outputs the two challenge messages from which one will be encrypted and given to the latter, which then tries to determine which message was encrypted. Both parts of the adversary have access to a random oracle H; in the classical game (INDCPA) access is restricted to be classical, in the post-quantum game ($\mathsf{pqINDCPA}$) superposition access is granted.

In Section 5.1.1 we define a special class of encryption schemes for which our lifting theorem is applicable. We classify two types of game hops for this class of encryption schemes in Section 5.1.2. Section 5.1.3 develops requirements under which proofs for the game hops also hold against quantum adversaries. In Section 5.1.4 we apply the lifting theorem to three public key encryption schemes.

Game INDCPA	Game $\mathsf{pqINDCPA}$
$b \leftarrow\!\!\$\ \{0,1\}$	$b \leftarrow\!\!\$\ \{0,1\}$
$(pk, sk) \leftarrow\!\!\$\ \mathtt{KGen}()$	$(pk, sk) \leftarrow\!\!\$\ \mathtt{KGen}()$
$m_0, m_1 \leftarrow\!\!\$\ \mathcal{A}_{\mathsf{m}}^{\mathsf{H}}(pk)$	$m_0, m_1 \leftarrow\!\!\$\ \mathcal{A}_{\mathsf{m}}^{\lvert\mathsf{H}\rangle}(pk)$
$c \leftarrow\!\!\$\ \mathtt{Enc}^{\mathsf{H}}(pk, m_b)$	$c \leftarrow\!\!\$\ \mathtt{Enc}^{\mathsf{H}}(pk, m_b)$
$b' \leftarrow\!\!\$\ \mathcal{A}_{\mathsf{d}}^{\mathsf{H}}(pk, c)$	$b' \leftarrow\!\!\$\ \mathcal{A}_{\mathsf{d}}^{\lvert\mathsf{H}\rangle}(pk, c)$
return $(b' = b)$	**return** $(b' = b)$

Figure 5.1: Security games INDCPA and $\mathsf{pqINDCPA}$ restricted to a single challenge for an oracle-simple PKE scheme $\textsc{Pke}$ using a random oracle H.

5.1.1 Requirements for PKE Schemes

We start by defining so-called *oracle-simple* public key encryption schemes. These are encryption schemes where the encryption algorithm invokes the random oracle exactly once on an input independent of the message and the public key.[26] Below we formally define such schemes.

Definition 5.1.1. *Let* $\textsc{Pke} = (\mathtt{KGen}, \mathtt{Enc}^{\mathsf{H}}, \mathtt{Dec}^{\mathsf{H}})$ *be a public key encryption scheme. If there exists an algorithm* $\mathtt{Enc\text{-}Sub}$ *and a deterministic function* $\mathtt{F}$ *which maps from some set* $\mathcal{R}$ *to* $\mathsf{Dom}(\mathsf{H})$ *such that* $\mathtt{Enc}^{\mathsf{H}}$ *can be written as in Figure 5.2, i.e., it first invokes the*

[26] This property is required to get a meaningful bound from applying the one-way to hiding lemma. Since we are not aware of any PKE scheme which does not satisfy this requirement, we do not consider it a restriction.

random oracle on $\mathtt{F}(r)$ *for a random* $r \in \mathcal{R}$ *to obtain* y *and then computes the ciphertext using* $\mathtt{Enc\text{-}Sub}(pk, m, r, y)$*, then we call* PKE *an oracle-simple (public key) encryption scheme with function* F *and encryption sub-routine* Enc-Sub*.*

$\mathtt{Enc}^{\mathsf{H}}(pk, m)$

$r \leftarrow_{\$} \mathcal{R}$
$x \leftarrow \mathtt{F}(r)$
$y \leftarrow \mathsf{H}(x)$
$c \leftarrow_{\$} \mathtt{Enc\text{-}Sub}(pk, m, r, y)$
return c

Figure 5.2: Algorithm Enc of an oracle-simple encryption scheme using F and Enc-Sub.

Based on this definition, we can rewrite the INDCPA and pqINDCPA security games for oracle-simple encryption schemes yielding the security games displayed in Figure 5.3. It corresponds to INDCPA as displayed in Figure 2.10. The difference is that we consider the single challenge case. There is no oracle LR-Enc, instead the adversary $\mathcal{A}$ is divided into two parts $\mathcal{A}_{\mathsf{m}}$ and $\mathcal{A}_{\mathsf{d}}$; the former corresponds to $\mathcal{A}$ up to the one query to LR-Enc whereas the latter corresponds to $\mathcal{A}$ up from the point of receiving the challenge ciphertext.

Since our lifting theorem is based on oracle-simple encryption schemes, its generality depends on the generality of this class of encryption schemes. Analysing all encryption schemes submitted as Round 1 NIST candidates which use random oracles [Ban19, AOP+17, ABD+17, CBB+17], reveals that all of them are indeed oracle-simple schemes. Thus, we see this as a style of notation which greatly simplifies the presentation of our proofs, rather than a restriction of its generality. Note that this analysis is based on the underlying encryption scheme as all candidates use random oracles when applying generic transformations to achieve CCA security.

5.1.2 Identification of Game Hops

Within this section we define two different types of game hops which are used to prove security of oracle-simple encryption schemes. Due to the structure of oracle-simple encryption schemes, we can distinguish between game hops for which lifting is rather trivial since they are independent of the random oracle, and game hops which are not independent of the random oracle. We start by defining a Type-I game hop which is independent of the random oracle.

Definition 5.1.2. *Let* G_i *and* G_{i+1} *be two* IND-CPA *games (cf. Figure 5.3) for an oracle-simple public key encryption scheme* $\text{PKE} = (\mathtt{KGen}, \mathtt{Enc}^{\mathsf{H}}, \mathtt{Dec}^{\mathsf{H}})$*. We call the game hop between* G_i *and* G_{i+1} *a* Type-I game hop *if the games only differ in using different algorithms* KGen *to generate the key pair or different algorithms* Enc-Sub *to generate the ciphertext.*

Next, we define a Type-II game hop which affects the usage of the random oracle while encrypting one of the challenge messages by the adversary.

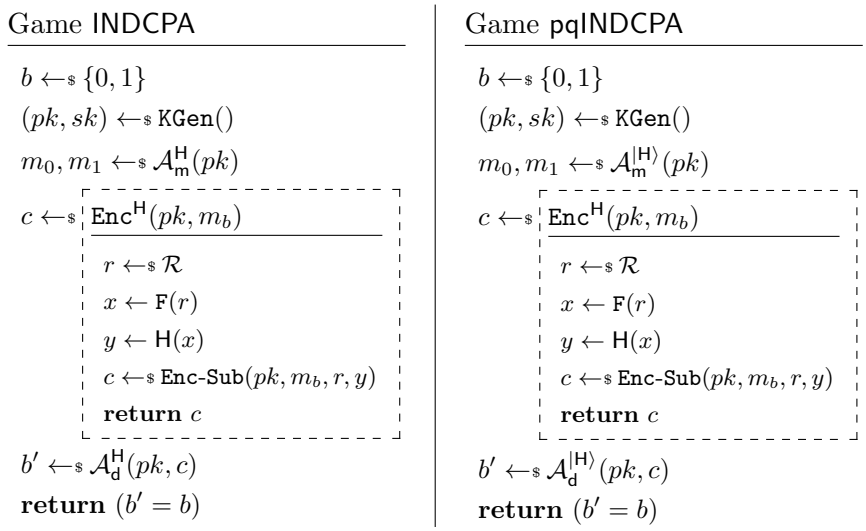

Figure 5.3: Security games INDCPA and pqINDCPA for an oracle-simple public key encryption scheme $\textsc{Pke} = (\mathtt{KGen}, \mathtt{Enc}^{\mathsf{H}}, \mathtt{Dec}^{\mathsf{H}})$ with function $\mathtt{F}$.

Definition 5.1.3. *Let* G_i *and* G_{i+1} *be two* IND-CPA *games (cf. Figure 5.3) for an oracle-simple public key encryption scheme* $\textsc{Pke} = (\mathtt{KGen}, \mathtt{Enc}^{\mathsf{H}}, \mathtt{Dec}^{\mathsf{H}})$*. We call the game hop between* G_i *and* G_{i+1} *a* Type-II game hop *if their only difference is that game* G_i *obtains* y *by invoking* H *on* x *while game* G_{i+1} *samples* y *uniformly at random from* $\mathsf{CoDom}(\mathsf{H})$*.*

Having discussed the generality of the class of encryption schemes, the next natural question asks for the generality of the defined game hops. A Type-II game hop is a standard game hop to make the challenge independent of the random oracle, thereby rendering it obsolete for the adversary. As for Type-I game hops, we observe the following. To bound the game advantage, one transforms an adversary that distinguishes the games into an adversary (the reduction) that solves some problem. To achieve this, the game hop has to be connected with the problem instance. Thus, the reduction has to feed the problem instance to the adversary. Considering IND-CPA security, its options are fairly limited. Either the reduction feeds its challenge via the inputs to the adversary—the public key pk or the ciphertext c—or as a response from the random oracle. The former case is the one we cover with a Type-I game hop. The latter case is not covered, as none of the schemes, that we are aware of, requires such a game hop. Nevertheless, we emphasise that our lifting theorem can be easily extended by another type of game hop, if needed. The post-quantum analogue of such a challenge injection in a random oracle response can be achieved using Zhandry's semi-constant distributions [Zha12b], where a challenge is injected in a subset of inputs which gives a significant chance that the adversary uses the injected challenge while the probability of detecting the challenge injection remains small enough.

5.1.3 Lifting Security

In this section we state the conditions under which a classical security proof holds true in the post-quantum setting.

The lemma below states that classical reductions from a decisional problem to the game advantage of a Type-I game hop hold true in the post-quantum setting.

Lemma 5.1.4. *Let* G_i *and* G_{i+1} *be games such that the game hop between these is a* Type-I *game hop. Suppose there exists a decisional problem* dP *which is reduced to the advantage of distinguishing between* G_i *and* G_{i+1}*. Then, for any quantum adversary* $\mathcal{A}$*, there exists a quantum adversary* $\mathcal{B}$ *against* dP *such that*

$$\left|\Pr[\mathsf{G}_i^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathsf{G}_{i+1}^{\mathcal{A}} \Rightarrow 1]\right| \leq \mathbf{Adv}^{\mathsf{dP}}(\mathcal{B})\,.$$

Proof. The difference between the games is independent from the random oracle. Hence the same proof holds against quantum adversaries, albeit adversary $\mathcal{B}$ has to simulate a quantum random oracle for adversary $\mathcal{A}$. This can be done using a $2q_{\mathsf{H}}$-wise independent function, where q_{H} is the number of random oracle queries made by $\mathcal{A}$. Zhandry [Zha12b] showed that this simulation is perfect in that the adversary cannot distinguish the simulated oracle from a real one. □

Alternatively, Lemma 5.1.4 can be formally proven using the framework by Song [Son14]. Due to the complex notation used in [Son14], however, this leads to a rather long and tedious proof.

The following lemma states conditions under which the classical proof for a Type-II game hop holds true against quantum adversaries.

Lemma 5.1.5. *Let* G_i *and* G_{i+1} *be games such that the game hop between these is a* Type-II *game hop. Suppose there exists a search problem* sP *which is reduced to the probability that an adversary queries the random oracle on* x*. Then, for any quantum adversary* $\mathcal{A}$*, making* q_{H} *queries to* $|\mathsf{H}\rangle$*, there exists a quantum adversary* $\mathcal{C}$ *against* sP *such that*

$$\left|\Pr[\mathsf{G}_i^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathsf{G}_{i+1}^{\mathcal{A}} \Rightarrow 1]\right| \leq 2q_{\mathsf{H}}\sqrt{\mathbf{Adv}^{\mathsf{sP}}(\mathcal{C})}\,.$$

Proof. We observe that the difference between games G_i and G_{i+1} lies in the random oracle. More precisely, in G_i the adversary gets access to the random oracle H; in G_{i+1} it gets access to the same random oracle albeit reprogrammed at input x. The only information that the adversary gets about x is via the ciphertext c. Thus we get

$$\left|\Pr[\mathsf{G}_i^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathsf{G}_{i+1}^{\mathcal{A}} \Rightarrow 1]\right| \leq \left|\Pr[\mathcal{A}^{\mathsf{H}}(c) \Rightarrow 1] - \Pr[\mathcal{A}^{\mathsf{H}_{x\mapsto\$}}(c) \Rightarrow 1]\right|\,. \tag{5.1}$$

We can apply the O2H lemma (cf. Lemma 2.6.1) to obtain

$$\left|\Pr[\mathcal{A}^{\mathsf{H}}(c) \Rightarrow 1] - \Pr[\mathcal{A}^{\mathsf{H}_{x\mapsto\$}}(c) \Rightarrow 1]\right| \leq 2q_{\mathsf{H}}\sqrt{\Pr[\mathcal{B}^{|\mathsf{H}\rangle}(c) \Rightarrow x]}\,, \tag{5.2}$$

where $\mathcal{B}$ is the adversary that simply runs $\mathcal{A}$ and outputs the measurement result of a randomly chosen query to H. At this point, we can use the classical proof which bounds

the probability of an adversary outputting x by the advantage of solving a search problem sP. This yields

$$2q_{\mathsf{H}}\sqrt{\Pr[\mathcal{B}^{|\mathsf{H}\rangle}(c) \Rightarrow x]} \leq 2q_{\mathsf{H}}\sqrt{\mathbf{Adv}^{\mathsf{sP}}(\mathcal{C})}\,. \tag{5.3}$$

Combining (5.1), (5.2), and (5.3) proves the claim. □

Now we are ready to state our lifting theorem to lift classical security proofs of oracle-simple public key encryption schemes to the post-quantum setting.

Theorem 5.1.6. *Let* $\textsc{Pke} = (\mathtt{KGen}, \mathtt{Enc}^{\mathsf{H}}, \mathtt{Dec}^{\mathsf{H}})$ *be an oracle-simple PKE scheme with function* $\mathtt{F}$ *according to Definition 5.1.1 and* $\mathcal{A}$ *be any quantum adversary. Suppose there exists a classical security proof using a sequence of games* $\mathsf{G}_0, \ldots, \mathsf{G}_k$*, where* G_0 *is the* INDCPA *game instantiated with* Pke*,* G_k *is constructed such that* $\mathbf{Adv}^{\mathsf{G}_k}(\mathcal{A}) = 0$*, and* k *is constant. Let* $i \in \{0, \ldots, k-1\}$ *be such that the game hop between* G_i *and* G_{i+1} *is a* Type-II *game hop. If*

1. *for any* $j \in \{0, \ldots, k-1\} \setminus \{i\}$*, the game hop between* G_j *and* G_{j+1} *is a* Type-I *game hop such that a quantum hard (decisional) problem* dP_j *is reduced to the game advantage between* G_{j-1} *and* G_j *and*
2. *there is some quantum hard (search) problem* sP_i *that is reduced to the probability of querying the random oracle* H *on* x*,*

then Pke *is* pqIND-CPA*-secure.*

Proof. The proof follows essentially by applying the previous lemmas. It holds that

$$\begin{aligned}
\mathbf{Adv}^{\mathsf{pqINDCPA}}_{\textsc{Pke}}(\mathcal{A}) &= \left|2\,\Pr[\mathsf{pqINDCPA}^{\mathcal{A}} \Rightarrow 1] - 1\right| \\
&= \left|2\,\Pr[\mathsf{G}_0^{\mathcal{A}} \Rightarrow 1] - 1\right| \\
&\leq 2\sum_{j=0}^{k-1}\left|\Pr[\mathsf{G}_j^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathsf{G}_{j+1}^{\mathcal{A}} \Rightarrow 1]\right| + \left|2\,\Pr[\mathsf{G}_k^{\mathcal{A}} \Rightarrow 1] - 1\right| \\
&= 2\sum_{j=0}^{k-1}\left|\Pr[\mathsf{G}_j^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathsf{G}_{j+1}^{\mathcal{A}} \Rightarrow 1]\right| + \mathbf{Adv}^{\mathsf{G}_k}(\mathcal{A})\,.
\end{aligned} \tag{5.4}$$

By applying Lemma 5.1.4, we obtain

$$\left|\Pr[\mathsf{G}_j^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathsf{G}_{j+1}^{\mathcal{A}} \Rightarrow 1]\right| \leq \mathbf{Adv}^{\mathsf{dP_j}}(\mathcal{B}_\mathsf{j})\,, \tag{5.5}$$

where $j \in [k]$. By applying Lemma 5.1.5, we get

$$\left|\Pr[\mathsf{G}_i^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathsf{G}_{i+1}^{\mathcal{A}} \Rightarrow 1]\right| \leq 2q_{\mathsf{H}}\sqrt{\mathbf{Adv}^{\mathsf{sP_i}}(\mathcal{C})}\,. \tag{5.6}$$

Finally, we have

$$\mathbf{Adv}^{\mathsf{G}_k}(\mathcal{A}) = 0\,. \tag{5.7}$$

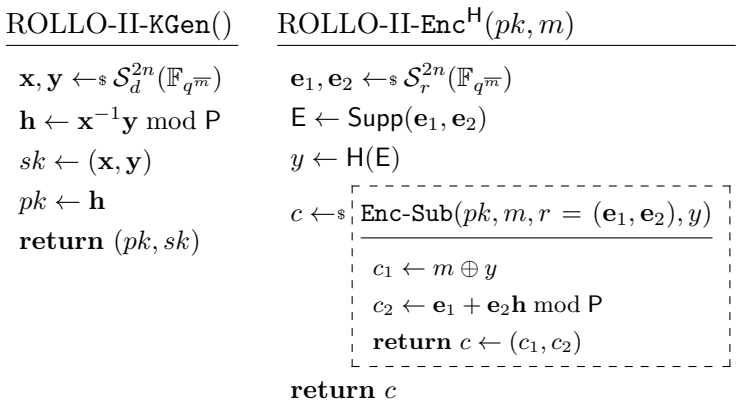

Figure 5.4: Encryption scheme ROLLO-II written as oracle-simple encryption scheme.

Inserting (5.5), (5.6), and (5.7) into (5.4) yields

$$\begin{aligned}\mathbf{Adv}^{\mathsf{pqINDCPA}}_{\mathrm{PKE}}(\mathcal{A}) &\leq 2 \sum_{j\in[k]} \left|\Pr[\mathsf{G}_j^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathsf{G}_{j+1}^{\mathcal{A}} \Rightarrow 1]\right| + \mathbf{Adv}^{\mathsf{G}_k}(\mathcal{A}) \\ &\leq 2 \sum_{j\in[k]\setminus\{i\}} \mathbf{Adv}^{\mathsf{dP_j}}(\mathcal{B}_\mathsf{j}) + 2q_\mathsf{H}\sqrt{\mathbf{Adv}^{\mathsf{sP_i}}(\mathcal{C})}\,.\end{aligned}$$

Using the fact that k is constant, all advantages are negligible against quantum adversaries, and the number of random oracle queries is polynomial yields

$$\mathbf{Adv}^{\mathsf{pqINDCPA}}_{\mathrm{PKE}}(\mathcal{A}) \leq 2 \sum_{j\in[k]\setminus\{i\}} \mathrm{negl}(\lambda) + 2q_\mathsf{H}\sqrt{\mathrm{negl}(\lambda)} \leq \mathrm{negl}(\lambda)\,.$$

This proves the theorem. □

5.1.4 Post-Quantum Security of PKE Schemes

We use our lifting theorem to lift the classical security of three public key encryption schemes to post-quantum security. We first lift the security for the code-based public key encryption scheme ROLLO-II [ABD+19], followed by the lattice-based public key encryption scheme LARA [Ban19]. Neither of these two schemes were proven secure in the QROM thus far.

Code-based Public Key Encryption Scheme ROLLO-II

To apply the lifting theorem to the code-based PKE scheme ROLLO-II, we first need to show that it is an oracle-simple PKE scheme. This is shown in Figure 5.4 which displays the scheme as an oracle-simple scheme. The security of ROLLO-II is based on the ideal rank support recovery (IRSR) problem and the ideal low rank parity check (ILRPC) codes indistinguishability problem. The theorem below shows that the code-based encryption scheme ROLLO-II is pqIND-CPA-secure, assuming that both problems are hard for quantum adversaries.

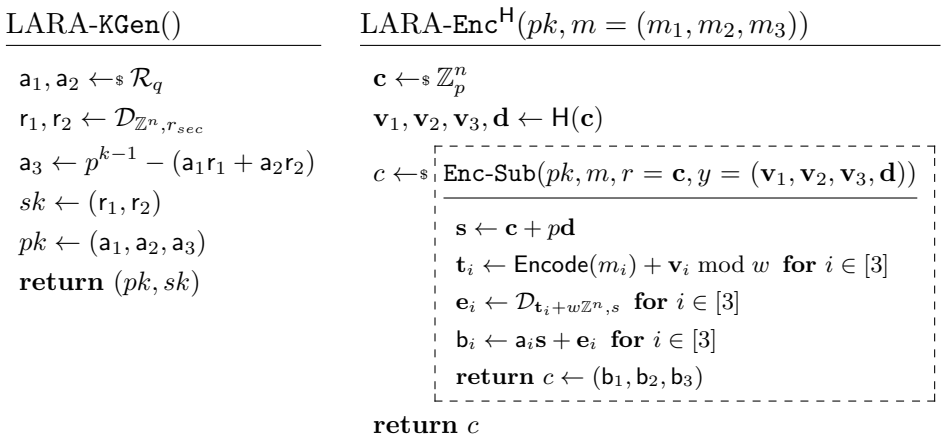

Figure 5.5: Encryption scheme LARA written as an oracle-simple encryption scheme.

Theorem 5.1.7. *Assuming the post-quantum hardness of the ILRPC problem (cf. Problem 2.6.10) and the IRSR problem (cf. Problem 2.6.9), the code-based encryption scheme ROLLO-II, described in Figure 2.30, is* pqIND-CPA*-secure.*

Proof. The classical IND-CPA security proof of ROLLO-II, given in [ABD+19], uses games $\mathsf{G}_0, \mathsf{G}_1, \mathsf{G}_2, \mathsf{G}_3$. Except for the first game G_0, we only state the change to its predecessor.

Game G_0: This is the game INDCPA instantiated with ROLLO-II.

Game G_1: In this game the vector $\mathbf{h}$ (the public key) is sampled randomly.

Game G_2: The value y is sampled randomly, independent of H.

Game G_3: The value c_1 is sampled randomly.

The game hop between G_1 and G_2 is a Type-II game hop, while all other game hops are Type-I game hops. The classical proof reduces the ILRPC problem (cf. Problem 2.6.10) to distinguishing games G_0 and G_1 (Type-I) and the IRSR problem (cf. Problem 2.6.9) to the probability of querying the random oracle on $\mathsf{E} = \mathsf{Supp}(\mathbf{e}_1, \mathbf{e}_2)$ and thereby also to distinguishing games G_1 and G_2 (Type-II). The game hop between G_2 and G_3 (Type-I) is bound by the problem of distinguishing between a one-time pad encryption and a random ciphertext. Since all these problems are assumed to be hard even for quantum adversaries, Theorem 5.1.6 proves the claim. □

Lattice-based Public Key Encryption Scheme LARA

In Figure 5.5 we show that LARA is an oracle-simple public key encryption scheme which allows to apply our lifting theorem. The security proof of LARA is based on the LWE problem, precisely, on some variants of it. Assuming these problems to be hard for quantum adversaries, the following theorem shows the pqIND-CPA security of LARA.

Theorem 5.1.8. *Assuming the post-quantum hardness of the DLWE problem (cf. Problem 2.6.7) and the LWE problem (cf. Problem 2.6.6), the lattice-based encryption scheme LARA, described in Figure 5.5, is* pqIND-CPA*-secure.*

Proof. The classical IND-CPA security proof of LARA, given in [Ban19], uses games $\mathsf{G}_0, \mathsf{G}_1, \mathsf{G}_2, \mathsf{G}_3, \mathsf{G}_4$. Except for game G_0, we only state the change to its predecessor.

Game G_0: This is the INDCPA game instantiated with LARA.

Game G_1: In this game the polynomial a_3 is sampled randomly.

Game G_2: The vectors $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3, \mathbf{d}$ are sampled randomly, independent of H.

Game G_3: The polynomials e_i are sampled according to the distribution $\mathcal{D}_{\mathbb{Z}^n,s}$.

Game G_4: The polynomials b_i are sampled randomly.

The game hop between G_1 and G_2 is a Type-II game hop, while all other game hops are Type-I game hops. The classical proof reduces the DLWE problem (cf. Problem 2.6.7), with a different number of samples, to distinguishing games Type-I game hops. The Type-II game hop is bound by the low-bit-LWE (lbLWE) problem which, in turn, is bound by the hardness of the LWE problem (cf. Problem 2.6.6). Thus, we can apply Theorem 5.1.6 which proves the claim. □

5.2 Post-Quantum Security of SLAE

In this section we show that the AEAD scheme SLAE, developed in Section 3.2, is post-quantum secure. In Section 5.2.1 we show that the post-quantum security boils down to the post-quantum security of the underlying components SLFUNC, SPRG, and SVHASH. Subsequently, we show in Section 5.2.2, Section 5.2.3, and Section 5.2.4 that SLFUNC, SPRG, and SVHASH, respectively, are post-quantum secure.

5.2.1 Security of SLAE

For SLAE, we target the two security notions pqIND-CPA (in the Left-or-Right sense) and INT-CTXT against quantum adversaries. Recall from Section 2.2.3, that these two notions together are equivalent to the combined security notion of AE.

In this section we show that the pqIND-CPA and INT-CTXT security of the authenticated encryption scheme SLAE against quantum adversaries follows from the post-quantum security of the underlying primitives SLFUNC, SPRG, and SVHASH. For this we first show that the security of SLFUNC and SPRG yields the pqIND-CPA security of SLENC, which, in turn, yields pqIND-CPA and INT-CTXT security of SLAE. We then show that the security of SLFUNC and SVHASH yields the SUF-CMA security of SLMAC which directly yields INT-CTXT security of SLAE.

pqIND-CPA **Security of** SLAE**.** The pqIND-CPA security follows from SLFUNC and SPRG being a secure PRF and PRG, respectively. Theorem 5.2.1 first shows that SLFUNC and SPRG being a secure PRF and secure PRG, respectively, yield pqIND-CPA security of the symmetric encryption scheme SLENC. Theorem 5.2.2 then establishes the pqIND-CPA security of SLAE based on the pqIND-CPA security of SLENC.

Theorem 5.2.1. *Let* SLFUNC *be a pseudorandom function and* SPRG *a pseudorandom generator. Let further* SLENC *be the symmetric key encryption scheme constructed from* SLFUNC *and* SPRG *as shown in Figure 3.9. For any quantum adversary $\mathcal{A}$ against the* pqIND-CPA *security, making q_{Enc} queries to its encryption oracle, there exist adversaries $\mathcal{A}_{\mathsf{prf}}$ and $\mathcal{A}_{\mathsf{prg}}$ against* SLFUNC *and* SPRG*, respectively, such that*

$$\mathbf{Adv}^{\mathsf{pqINDCPA}}_{\mathrm{SLENC}}(\mathcal{A}) \leq 2\,\mathbf{Adv}^{\mathsf{PRF}}_{\mathrm{SLFUNC}}(\mathcal{A}_{\mathsf{prf}}) + 2q\,\mathbf{Adv}^{\mathsf{PRG}}_{\mathrm{SPRG}}(\mathcal{A}_{\mathsf{prg}})\,.$$

Proof. The proof proceeds in two game hops. The first game hop replaces the function SLFUNC by a random function which can be straightforwardly bound by the PRF advantage of SLFUNC. More precisely, $\mathcal{A}_{\mathsf{prf}}$ uses its own oracle for everything related to SLFUNC while simulating SPRG using (classical) queries to the random oracle ρ. All (quantum) queries by $\mathcal{A}$ to ρ are simply forwarded by $\mathcal{A}_{\mathsf{prf}}$, as are the responses back to $\mathcal{A}$.

The second game hop replaces the output of SPRG by a random output. A standard hybrid argument [FM21] shows that this can be bound by the security of SPRG. The reduction $\mathcal{A}_{\mathsf{prg}}$ picks a random query of $\mathcal{A}$ to its encryption oracle, where it uses its own input (either the output of SPRG or a random bit string) to encrypt the message. Prior queries are answered by XORing random bit string to the message while subsequent queries are answered by simulating SPRG using (classical) queries to ρ. All (quantum) queries by $\mathcal{A}$ to ρ are simply forwarded by $\mathcal{A}_{\mathsf{prg}}$, as are the responses back to $\mathcal{A}$.

The resulting game yields identically distributed ciphertexts, irrespectively of the message. The factor 2 accounts for doing the game hops for both cases $b = 0$ and $b = 1$. □

Theorem 5.2.2. *Let* SLENC *be the symmetric key encryption scheme and* SLMAC *be a MAC. Further, let* SLAE *be the authenticated encryption scheme constructed from* SLENC *and* SLMAC *as shown in Figure 3.9. For any quantum adversary $\mathcal{A}$ against the* pqIND-CPA *security, making q_{Enc} queries to its encryption oracle, there exists an adversary $\mathcal{A}_{\mathsf{se}}$, such that*

$$\mathbf{Adv}^{\mathsf{pqINDCPA}}_{\mathrm{SLAE}}(\mathcal{A}) \leq \mathbf{Adv}^{\mathsf{pqINDCPA}}_{\mathrm{SLENC}}(\mathcal{A}_{\mathsf{se}})\,.$$

Proof. The proof proceeds by a simple reduction. In more detail, the reduction $\mathcal{A}_{\mathsf{se}}$ picks a key for the MAC SLMAC. For every query to the encryption oracle by $\mathcal{A}$, $\mathcal{A}_{\mathsf{se}}$ invokes its own encryption oracle and locally computes the tag of the ciphertext using (classical) queries to ρ before sending both the ciphertext and the tag back to $\mathcal{A}$. Every (quantum) query by $\mathcal{A}$ to ρ is simply forwarded by $\mathcal{A}_{\mathsf{se}}$. □

INT-CTXT **Security of** SLAE**.** The INT-CTXT security follows from SLFUNC being a secure PRF and SVHASH being a collision-resistant hash function. In Theorem 5.2.3, we show that both yield a SUF-CMA-secure MAC SLMAC. Subsequently, Theorem 5.2.4 shows that the SUF-CMA security of SLMAC ensures INT-CTXT security of SLAE.

Theorem 5.2.3. *Let* SLFUNC *be a function and* SVHASH *a hash function. Let further* SLMAC *be the MAC constructed from* SLFUNC *and* SVHASH *as shown in Figure 3.9. For any quantum adversary $\mathcal{A}$ against the* SUF-CMA *security, making q_T queries to its tagging*

oracle and q_F to its forge oracle, there exist adversaries $\mathcal{A}_{\mathsf{prf}}$ and $\mathcal{A}_{\mathsf{hash}}$ against SLFUNC *and* SVHASH*, respectively, such that*

$$\mathbf{Adv}^{\mathsf{SUFCMA}}_{\text{SLMAC}}(\mathcal{A}) \leq \mathbf{Adv}^{\mathsf{PRF}}_{\text{SLFUNC}}(\mathcal{A}_{\mathsf{prf}}) + \mathbf{Adv}^{\mathsf{CR}}_{\text{SVHASH}}(\mathcal{A}_{\mathsf{hash}}) + \frac{q_F}{2^\tau}\,.$$

Proof. We assume that all messages queried by $\mathcal{A}$ result in different hash values, otherwise, we obtain a simple reduction $\mathcal{A}_{\mathsf{hash}}$ from the collision resistance of SVHASH.

Then the proof proceeds by a game hop in which SLFUNC is replaced by a random function. The reduction $\mathcal{A}_{\mathsf{prf}}$ will invoke its own function to simulate the tagging and verification of SLMAC and (classical) queries to ρ to evaluate SVHASH. Every (quantum) query to ρ by $\mathcal{A}$ is simply forwarded by $\mathcal{A}_{\mathsf{prf}}$.

The resulting game is bound by a simple counting argument that $\mathcal{A}$ predicts the output of a random function. □

Theorem 5.2.4. *Let* SLENC *be the symmetric key encryption scheme and* SLMAC *be a MAC. Let further* SLAE *be the authenticated encryption scheme constructed from* SLENC *and* SLMAC *as shown in Figure 3.9. For any quantum adversary $\mathcal{A}$ against the* INT-CTXT *security, making q_E queries to its encryption oracle and q_F queries to its forge oracle, there exists an adversary $\mathcal{A}_{\mathsf{mac}}$, such that*

$$\mathbf{Adv}^{\mathsf{INTCTXT}}_{\text{SLAE}}(\mathcal{A}) \leq \mathbf{Adv}^{\mathsf{SUFCMA}}_{\text{SLMAC}}(\mathcal{A}_{\mathsf{mac}})\,.$$

Proof. The reduction $\mathcal{A}_{\mathsf{mac}}$ picks a key for the symmetric key encryption scheme SLENC. For every query to the encryption oracle by $\mathcal{A}$, $\mathcal{A}_{\mathsf{mac}}$ locally computes the ciphertext using (classical) queries to ρ and obtains the tag using its own tagging oracle. Then, it sends the ciphertext and the tag back to $\mathcal{A}$. For every forgery attempt by $\mathcal{A}$, $\mathcal{A}_{\mathsf{mac}}$ queries the ciphertext and the tag as its own forgery attempt. If the tag verifies correctly, $\mathcal{A}_{\mathsf{mac}}$ locally decrypts the ciphertext using the sampled key and (classical) queries to ρ and sends the message back to $\mathcal{A}$, otherwise, i.e., if the tag was invalid, $\mathcal{A}_{\mathsf{mac}}$ simply returns $\perp$ to $\mathcal{A}$. Every (quantum) query by $\mathcal{A}$ to ρ is simply forwarded by $\mathcal{A}_{\mathsf{se}}$. □

5.2.2 Security of SLFUNC

The sponge-based pseudorandom function SLFUNC is illustrated in Figure 5.6[27], while the pseudocode can be found in Figure 3.10. The function initialises the state of the sponge with the key and then absorbs the input, in case of SLAE the nonce N, r bits at a time. After the nonce has been absorbed, the output is obtained by applying the transformation ρ a final time and outputting the state. Note that the function outputs the full state rather than squeezing it r-bit at a time over several rounds. That is also the reason why ρ is required to be a random transformation rather than a random permutation. Otherwise, an adversary could simply undo the transformation from the output by applying the inverse permutation. The theorem below gives a bound on distinguishing SLFUNC from a random function when having superposition access to the underlying random oracle ρ. The proof utilises the O2H lemma (cf. Lemma 2.6.1).

[27] The figure is identical to Figure 3.11 in Chapter 3, we display it here again for convenience. The same holds for Figure 5.7 and Figure 5.8.

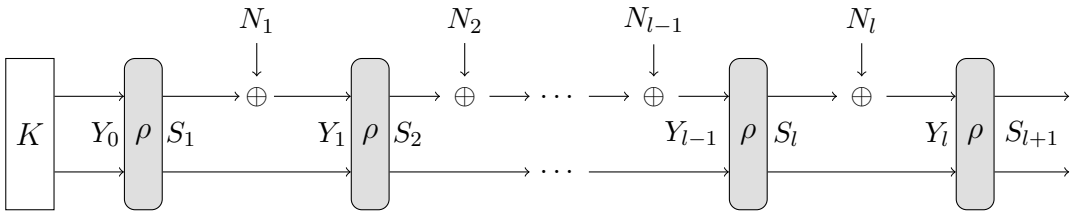

Figure 5.6: Visualisation of the sponge-based function SLFUNC.

Theorem 5.2.5. *Let $\mathcal{F} = \text{SLFUNC}$ be the function displayed in Figure 5.6. Then, for any quantum adversary $\mathcal{A}$, making $q_{\mathcal{F}}$ (classical) queries to* SLFUNC *and q_ρ (quantum) queries to ρ, it holds that*

$$\mathbf{Adv}^{\mathsf{PRF}}_{\text{SLFUNC}}(\mathcal{A}) \leq \frac{q_{\mathcal{F}}}{2^c} + 2q\sqrt{\frac{2^\nu}{2^n}}\,.$$

Proof. Let $l = \left\lceil \frac{\nu}{r} \right\rceil$ be the number of absorption steps and we assume for simplicity that ν is a multiple of the rate. We further recursively define sets $\mathcal{Y}_i$ as

$$\begin{aligned}\mathcal{Y}_0 &= \{K\}\\ \mathcal{Y}_i &= \{R \parallel \lfloor \rho(x) \rfloor_c \,|\, R \in \{0,1\}^r, x \in \mathcal{Y}_{i-1}\}\end{aligned}$$

for all $i \in \{0,\ldots,l\}$, i.e., $\mathcal{Y}_i$ is the set of all possible values that can occur as input to the i^{th} call to ρ while evaluating $\mathcal{F}(K,\cdot)$. It follows that $|\mathcal{Y}_i| = 2^{ir}$ and, in particular, $|\mathcal{Y}_l| = 2^{lr} = 2^\nu$. Note that every input N defines a sequence of states $Y_0, Y_1, \ldots, Y_l$ that occur while evaluating the sponge. For an input N, let $Y_i[N]$ denote the state Y_i for this particular input, e.g., $Y_1[N] = (\lceil \rho(K) \rceil^r \oplus N_1) \parallel \lfloor \rho(K) \rfloor_c$, where $N = N_1 \parallel \ldots \parallel N_l$ with $|N_i| = r$. In particular, for every input N it holds that $Y_0[N] = K$.

We want to bound the following difference

$$\mathbf{Adv}^{\mathsf{PRF}}_{\text{SLFUNC}}(\mathcal{A}) = \left| \Pr_{K \leftarrow\$ \mathcal{K}}[\mathcal{A}^{\mathcal{F}(K,\cdot),\rho} \Rightarrow 1] - \Pr_{\mathcal{F}^* \leftarrow\$ \mathcal{Y}^{\mathcal{X}}}[\mathcal{A}^{\mathcal{F}^*(\cdot),\rho} \Rightarrow 1] \right| .$$

In order to do this, we define the oracle ρ^*, where $\rho^*(Y_l[N]) = \mathcal{F}^*(N)$ for all $Y_l[N] \in \mathcal{Y}_l$ and $\rho^*(x) = \rho(x)$ else. That is, oracle ρ^* is reprogrammed on all final input states $Y_l[N]$ to output the output of the random function $\mathcal{F}^*$ on input N. Then it holds that

$$\begin{aligned}\mathbf{Adv}^{\mathsf{PRF}}_{\text{SLFUNC}}(\mathcal{A}) &= \left| \Pr_{K \leftarrow\$ \mathcal{K}}[\mathcal{A}^{\mathcal{F}(K,\cdot),\rho} \Rightarrow 1] - \Pr_{\mathcal{F}^* \leftarrow\$ \mathcal{Y}^{\mathcal{X}}}[\mathcal{A}^{\mathcal{F}^*(\cdot),\rho} \Rightarrow 1] \right|\\ &\leq \left| \Pr_{K \leftarrow\$ \mathcal{K}}[\mathcal{A}^{\mathcal{F}(K,\cdot),\rho} \Rightarrow 1] - \Pr_{\mathcal{F}^* \leftarrow\$ \mathcal{Y}^{\mathcal{X}}}[\mathcal{A}^{\mathcal{F}^*(\cdot),\rho^*} \Rightarrow 1] \right|\\ &\quad + \left| \Pr_{\mathcal{F}^* \leftarrow\$ \mathcal{Y}^{\mathcal{X}}}[\mathcal{A}^{\mathcal{F}^*(\cdot),\rho^*} \Rightarrow 1] - \Pr_{\mathcal{F}^* \leftarrow\$ \mathcal{Y}^{\mathcal{X}}}[\mathcal{A}^{\mathcal{F}^*(\cdot),\rho} \Rightarrow 1] \right| .\end{aligned}$$

For the first difference on the right-hand, the oracles are consistent in both cases. However, if the adversary finds a collision on the final input to ρ for $\text{SLFUNC}(K,\cdot)$, more precisely,

two inputs N and N' such that $\lceil N \rceil^{\nu-r} \neq \lceil N' \rceil^{\nu-r}$ and $Y_l[N] = Y_l[N']$, then these two input will result in the same output for $\mathcal{F}$ and (most likely) different outputs for $\mathcal{F}^*$. Bounding the probability of finding such a collision is a counting argument over the number of queries to the function and it follows that

$$\left| \Pr_{K \leftarrow\$ \mathcal{K}}[\mathcal{A}^{\mathcal{F}(K,\cdot),\rho} \Rightarrow 1] - \Pr_{\mathcal{F}^* \leftarrow\$ \mathcal{Y}^{\mathcal{X}}}[\mathcal{A}^{\mathcal{F}^*(\cdot),\rho^*} \Rightarrow 1] \right| \leq \frac{q_{\mathcal{F}}}{2^c} \,.$$

For the second difference, we can apply the O2H lemma (cf. Lemma 2.6.1) which yields

$$\left| \Pr_{\mathcal{F}^* \leftarrow\$ \mathcal{Y}^{\mathcal{X}}}[\mathcal{A}^{\mathcal{F}^*(\cdot),\rho^*} \Rightarrow 1] - \Pr_{\mathcal{F}^* \leftarrow\$ \mathcal{Y}^{\mathcal{X}}}[\mathcal{A}^{\mathcal{F}^*(\cdot),\rho} \Rightarrow 1] \right| \leq 2q_\rho \sqrt{\Pr[x \in \mathcal{Y}_l \,|\, \mathcal{B}^{\mathcal{F}^*(\cdot),\rho} \Rightarrow x]} \,.$$

Recall that $\mathcal{B}^{\mathcal{F}^*(\cdot),\rho}$ simply runs $\mathcal{A}^{\mathcal{F}^*(\cdot),\rho}$ and outputs the measurement outcome of a randomly chosen query to ρ. However, $\mathcal{A}$ has no information about the set $\mathcal{Y}_l$, hence, we conclude with

$$2\, q_\rho \sqrt{\Pr[x \in \mathcal{Y}_l \,|\, \mathcal{B}^{\mathcal{F}^*(\cdot),\rho} \Rightarrow x]} \leq 2q_\rho \sqrt{\frac{|\mathcal{Y}_l|}{2^n}} \leq 2q_\rho \sqrt{\frac{2^{lr}}{2^n}} \leq 2q_\rho \sqrt{\frac{2^\nu}{2^n}} \,.$$

Collecting everything yields

$$\mathbf{Adv}^{\mathsf{PRF}}_{\mathrm{SLF}_{\mathrm{UNC}}}(\mathcal{A}) = \left| \Pr_{K \leftarrow\$ \mathcal{K}}[\mathcal{A}^{\mathcal{F}(K,\cdot),\rho} \Rightarrow 1] - \Pr_{\mathcal{F}^* \leftarrow\$ \mathcal{Y}^{\mathcal{X}}}[\mathcal{A}^{\mathcal{F}^*(\cdot),\rho} \Rightarrow 1] \right| \leq \frac{q_{\mathcal{F}}}{2^c} + 2q_\rho \sqrt{\frac{2^\nu}{2^n}} \,.$$

This concludes the proof. □

5.2.3 Security of SPRG

In this section we show that the sponge-based pseudorandom generator SPRG is secure against adversaries having superposition access to the underlying random oracle ρ. The PRG SPRG is displayed in Figure 5.7 and the respective pseudocode is given in Figure 3.10. The construction deviates from more common constructions for pseudorandom generators since it initialises the state of the sponge with the seed rather than absorbing it. The output is then generated by squeezing r bits at each iteration of the sponge. Similar to the previous section, the proof relies on the O2H lemma.

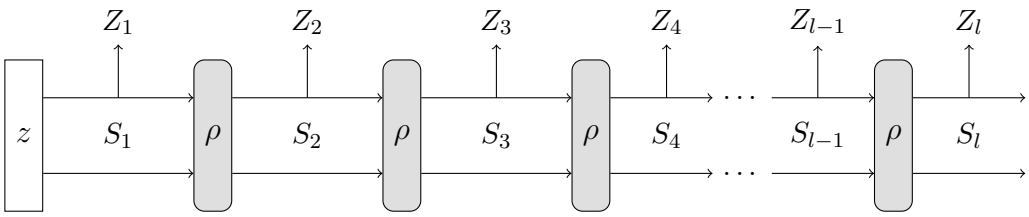

Figure 5.7: Visualisation of the sponge-based pseudorandom generator SPRG.

Theorem 5.2.6. *Let* SPRG *be the pseudorandom generator displayed in Figure 5.7. Then for any quantum adversary $\mathcal{A}$, making q (quantum) queries to ρ, and receiving an input of length μ it holds that*

$$\mathbf{Adv}^{\mathsf{PRG}}_{\mathrm{SPRG}}(\mathcal{A}) \leq \frac{2lq}{\sqrt{2^c}}\,,$$

where $l = \left\lceil \frac{\mu}{r} \right\rceil$ is the number of squeezing steps to obtain the required output length μ.

Proof. We assume, for sake of simplicity, that μ is a multiple of r. For a seed z, let $S_1, S_2, \ldots, S_l$ denote the sequence of states that occur during evaluation of the sponge, i.e., $S_i = \rho^{i-1}(z)$, where ρ^i corresponds to i consecutive evaluations of ρ. We want to bound the following difference

$$\mathbf{Adv}^{\mathsf{PRG}}_{\mathrm{SPRG}}(\mathcal{A}) = \left| \Pr_{z \leftarrow\$\, \mathcal{S}}[\mathcal{A}^{\rho}(Z) \Rightarrow 1] - \Pr_{R \leftarrow\$\, \{0,1\}^{\mu}}[\mathcal{A}^{\rho}(R) \Rightarrow 1] \right|,$$

where $Z = Z_1 \parallel \cdots \parallel Z_l = \mathrm{SPRG}(z, lr)$, i.e., obtaining an output of length lr using SPRG on seed z and $R = R_1 \parallel \cdots \parallel R_l$, such that $|Z_i| = |R_i| = r$. We write $R_{[i,j]}$ for $R_i \parallel \cdots \parallel R_j$, the same for Z. In particular, $R_{[i,j]}$ for $i > j$ equals the empty string. We obtain

$$\begin{aligned}\mathbf{Adv}^{\mathsf{PRG}}_{\mathrm{SPRG}}(\mathcal{A}) &= \left|\Pr[\mathcal{A}^{\rho}(Z_{[1,l]}) \Rightarrow 1] - \Pr[\mathcal{A}^{\rho}(R_{[1,l]}) \Rightarrow 1]\right| \\ &\leq \sum_{i=1}^{l} \left|\Pr[\mathcal{A}^{\rho}(R_{[1,i-1]} \parallel Z_{[i,l]}) \Rightarrow 1] - \Pr[\mathcal{A}^{\rho}(R_{[1,i]} \parallel Z_{[i+1,l]}) \Rightarrow 1]\right|.\end{aligned}$$

We start with the first difference, that, after simple rewriting, is,

$$\begin{aligned}&\left|\Pr[\mathcal{A}^{\rho}(Z_1 \parallel Z_{[2,l]}) \Rightarrow 1] - \Pr[\mathcal{A}^{\rho}(R_1 \parallel Z_{[2,l]}) \Rightarrow 1]\right| \\ &\leq \left|\Pr[\mathcal{A}^{\rho}(Z_1 \parallel Z_{[2,l]}) \Rightarrow 1] - \Pr[\mathcal{A}^{\rho^1}(R_1 \parallel Z_{[2,l]}) \Rightarrow 1]\right| \\ &\quad + \left|\Pr[\mathcal{A}^{\rho^1}(R_1 \parallel Z_{[2,l]}) \Rightarrow 1] - \Pr[\mathcal{A}^{\rho}(R_1 \parallel Z_{[2,l]}) \Rightarrow 1]\right|,\end{aligned}$$

where $\rho^1(R_1 \parallel \lfloor S_1 \rfloor_c) = S_2$. Then, it holds that the first difference above is 0, as the relation between R_1 and ρ^1 is the same as between Z_1 and ρ, and we merely need to bound the second difference, which only differs in the random oracle (ρ and ρ^1) at input $R_1 \parallel \lfloor S_1 \rfloor_c$. Let $\mathcal{S}_1 = \{R_1 \parallel \lfloor S_1 \rfloor_c\}$, then we can apply the O2H lemma (cf. Lemma 2.6.1) to obtain

$$\begin{aligned}&\left|\Pr[\mathcal{A}^{\rho^1}(R_1 \parallel Z_{[2,l]}) \Rightarrow 1] - \Pr[\mathcal{A}^{\rho}(R_1 \parallel Z_{[2,l]}) \Rightarrow 1]\right| \\ &\qquad\qquad \leq 2q\sqrt{\Pr[x \in \mathcal{S}_1 \,|\, \mathcal{B}^{\rho}(R_1 \parallel Z_{[2,l]}) \Rightarrow x]}\,.\end{aligned}$$

While $\mathcal{A}$ knows R_1, it has no information about $\lfloor S_1 \rfloor_c$. Note that Z_i, for $i > 1$, provides no information about $\mathcal{S}_1$ due to ρ being one-way in the random oracle model. This yields

$$\Pr[x \in \mathcal{S}_1 \,|\, \mathcal{B}^{\rho}(R_1 \parallel Z_{[2,l]}) \Rightarrow x] \leq \frac{|\mathcal{S}_1|}{2^c} \leq \frac{1}{2^c}\,.$$

The same argument applies to the other differences, where more and more r-bit blocks of $\mathcal{A}$'s input are replaced with R_i. More precisely, we obtain

$$\begin{aligned}
&\left|\Pr[\mathcal{A}^{\rho}(R_{[1,i-1]} \parallel Z_{[i,l]}) \Rightarrow 1] - \Pr[\mathcal{A}^{\rho}(R_{[1,i]} \parallel Z_{[i+1,l]}) \Rightarrow 1]\right| \\
&\quad \leq 2q\sqrt{\Pr[x \in \mathcal{S}_i \,|\, \mathcal{B}^{\rho}(R_{[1,i]} \parallel Z_{[i+1,l]}) \Rightarrow x]} \\
&\quad \leq \frac{2q}{\sqrt{2^c}}\,,
\end{aligned}$$

where $\mathcal{S}_i = \{R_i \parallel \lfloor S_i \rfloor_c\}$. Collecting everything then yields

$$\begin{aligned}
\mathbf{Adv}^{\mathsf{PRG}}_{\mathrm{SPRG}}(\mathcal{A}) &= \left|\Pr[\mathcal{A}^{\rho}(Z_{[1,l-1]}) \Rightarrow 1] - \Pr[\mathcal{A}^{\rho}(R_{[1,l-1]}) \Rightarrow 1]\right| \\
&\leq \sum_{i=1}^{l} \left|\Pr[\mathcal{A}^{\rho}(R_{[1,i-1]} \parallel Z_{[i,l]}) \Rightarrow 1] - \Pr[\mathcal{A}^{\rho}(R_{[1,i]} \parallel Z_{[i+1,l]}) \Rightarrow 1]\right| \\
&\leq \sum_{i=1}^{l} 2q\sqrt{\Pr[x \in \mathcal{S}_i \,|\, \mathcal{B}^{\rho}(R_{[1,i]} \parallel Z_{[i+1,l]}) \Rightarrow x]} \\
&\leq \sum_{i=1}^{l} \frac{2q}{\sqrt{2^c}} \\
&\leq \frac{2lq}{\sqrt{2^c}}\,.
\end{aligned}$$

This concludes the proof. □

5.2.4 Security of SvHash

In this section we analyse the security of SvHash against adversaries having superposition access to the underlying random oracle ρ. We display SvHash in Figure 5.8 while its respective pseudocode can be found in Figure 3.10. Observe that in order to compute a hash digest, the internal state is initialised to an evaluation of the random transformation of a bit string of length n containing only zeros. Afterwards, the padded associated data and padded ciphertext are absorbed block-wise. To separate the boundary between associated data and ciphertext, the bit string $1 \parallel 0^{c-1}$ is XORed to the inner state $\hat{S}$ as soon as the associated data has been absorbed. Observe that the domain separation can be viewed as a sponge construction with a rate increased by one bit. In this sense, an adversary $\mathcal{A}$ against SvHash with rate r and capacity c can be viewed as an adversary against the plain sponge-based hash function with rate $r+1$ and capacity $c-1$, where $\mathcal{A}$ guarantees that the $(r+1)^{\text{th}}$ bit of each input block is 0 except for the block which corresponds to absorbing the first ciphertext block. Hence, a bound for the plain sponge-based hash function directly yields a bound for SvHash by accounting for the one-bit loss in the capacity.

Theorem 5.2.7. *Let* SvHash *be the hash function as displayed in Figure 5.8. Then, for any quantum adversary $\mathcal{A}$ making q (quantum) queries to ρ, it holds that*

$$\mathbf{Adv}^{\mathsf{CR}}_{\mathrm{SvHash}}(\mathcal{A}) \leq \sqrt{\epsilon_1} + l \cdot \epsilon_2 + \epsilon_3\,,$$

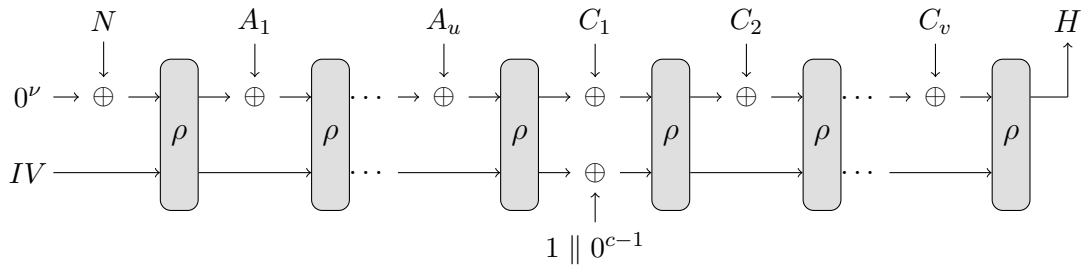

Figure 5.8: Visualisation of the sponge-based hash function SvHash.

where $\epsilon_1 \leq \frac{(q+1)^2}{2^{c-4}}$, $\epsilon_2 \leq q^3\left(\frac{\delta'+324}{2^{c-1}}\right) + 7\delta\sqrt{\frac{3(q+4)^3}{2^c}}$ *and* $\epsilon_3 \leq q^3\left(\frac{\delta'+324}{2^{w+1}}\right) + 7\delta\sqrt{\frac{3(q+4)^3}{2^{w+2}}}$ *with non-zero constants* δ *and* δ' *as well as* $l = \lceil \frac{\mu}{r} \rceil$ *where* μ *is the length of the (padded) message.*

Proof. The above collision resistance bound can be obtained from a combination of results from Czajkowski et al. [CBH+17] and Unruh [Unr15a] with a slight modification that stems from the way SvHash is constructed. Observe that the small modification is due to the interpretation that we consider a sponge-based hash function with the capacity being reduced by one bit and hence the rate being increased by one bit. We take care of this one-bit loss when applying the following results.

A crucial property in the realm of hash functions in the post-quantum setting is called the collapsing property which is a strengthening of collision resistance and Unruh has showed in [Unr15a, Unr16] that if a hash function is collapsing then this also implies that it is quantum collision-resistant. Additionally, Czajkowski et al. [CBH+17, CBH+18] showed that if the underlying function of the sponge construction is a random transformation then the sponge construction is collapsing. Being equipped with their result, we can derive the required bound for our setting.

We follow the proof strategy by Czajkowski et al. [CBH+17, Theorem 33] to show that the sponge construction is collapsing. This requires to show that the inner state $\hat{S}$ is collapsing in the absorbing phase while the outer state $\bar{S}$ is collapsing in the squeezing phase and that there are no zero-preimages in the inner state $\hat{S}$. Then using [Unr15a, Lemma 25] provides us with the implication that the sponge construction is then also collision-resistant. It now remains to apply the above strategy appropriately to derive the bound.

It holds that $l = \lceil \frac{\mu}{r} \rceil$ and by [CBH+17, Theorem 33], we know that the collapsing advantage is bounded by $\sqrt{\epsilon_1} + l \cdot \epsilon_2 + \epsilon_3$, where ϵ_1 corresponds to the probability of finding zero-preimages, ϵ_2 corresponds to the collapsing advantage of the inner state, and ϵ_3 corresponds to the collapsing advantage of the outer state. By applying [CBH+17, Lemma 19], we obtain that

$$\epsilon_1 \leq \frac{(q+1)^2}{2^{c-4}}\,.$$

By a simple combination of [CBH+17, Lemma 32] and [Unr15a, Theorem 38], we can

derive

$$\epsilon_2 \leq q^3 \left(\frac{\delta' + 324}{2^{c-1}} \right) + 7\delta \sqrt{\frac{3(q+4)^3}{2^c}}$$

and

$$\epsilon_3 \leq q^3 \left(\frac{\delta' + 324}{2^{w+1}} \right) + 7\delta \sqrt{\frac{3(q+4)^3}{2^{w+2}}},$$

where both δ and δ' are non-zero constants. Then, by [Unr15a, Lemma 25], we have a tight reduction from collapsing to collision resistance and hence the same bound holds for the collision resistance of the sponge construction. □

5.3 Post-Quantum Security of Deterministic Wallets

In recent years blockchain technologies received more and more attention. Besides basic primitives like public key encryption and signature schemes, quantum computers also threaten the security of blockchain protocols. This led to constructions for blockchain protocols based on post-quantum hardness assumptions [EZS+19, ESZ21, TMM21, EEE20].

In this section, we focus on deterministic wallets. Intuitively, these are wallets split into two entities that are able to derive session keys without communication, thereby increasing security as the wallet containing the secret key can stay offline unless a transaction is made. We show that an existing generic construction, proven secure in the ROM, achieves post-quantum security in the QROM.

5.3.1 Generic Wallet Construction

In this section, we recall the formal definition of a stateful wallet, following [DFL19], and the security properties that we want to guarantee for such a wallet. A stateful deterministic wallet scheme consists of two entities, a cold wallet and a hot wallet, that can derive a valid pair of secret and public keys, respectively, without the need for any interaction among each other. In more detail, upon initialization of the scheme, the cold wallet generates a master key pair (mpk, msk) and some initial state information st_0 and forwards (mpk, st_0) to the hot wallet. After this initial setup, the idea is that an arbitrary number of valid session key pairs can be generated by using the session public/secret key derivation algorithms within the respective wallets without further interaction. More precisely, the public key derivation algorithm takes as input the master public key and the current state to generate a session public key; the secret key derivation takes as inputs the master secret key and the current state in order to generate a session secret key. Since both public key and secret key derivation algorithms are deterministic, and the two wallets share the same current state, the key derivation algorithms output a valid session key pair. In order to keep track of which key has been derived with which state, each session key is indexed by a parameter id, which is given as input into the key derivation procedures. In the following we recall the definition of a deterministic stateful wallet scheme.

Definition 5.3.1. *A* stateful wallet scheme $\mathtt{SW}$ *is a tuple of five efficient algorithms* $(\mathtt{SW.KGen}, \mathtt{SW.RandSK}, \mathtt{SW.RandPK}, \mathtt{SW.Sign}, \mathtt{SW.Verify})$ *such that:*

- $\mathtt{SW.KGen}\colon \mathbb{N} \to \mathcal{PK} \times \mathcal{SK} \times \mathcal{ST}$ *is the key generation algorithm which takes a security parameter* λ *and returns a keypair consisting of a master public key* mpk *and a master secret key* msk *along with a state* st*.*
- $\mathtt{SW.RandSK}\colon \mathcal{SK} \times \mathcal{ID} \times \mathcal{ST} \to \mathcal{SK} \times \mathcal{ST}$ *is the secret key derivation algorithm that takes a master secret key* msk*, an identity* id*, and a state* st *as input to output a session secret key* sk_{id} *and a state* st*.*
- $\mathtt{SW.RandPK}\colon \mathcal{PK} \times \mathcal{ID} \times \mathcal{ST} \to \mathcal{PK} \times \mathcal{ST}$ *is the public key derivation algorithm that takes a master public key* mpk*, an identity* id*, and a state* st *as input to output a session public key* pk_{id} *and a state* st*.*
- $\mathtt{SW.Sign}\colon \mathcal{SK} \times \mathcal{PK} \times \mathcal{M} \to \mathcal{S}$ *is the signing algorithm which takes a session secret key* sk_{id}*, a session public key* pk_{id}*, and a message* m *as input and outputs a signature* σ*.*
- $\mathtt{SW.Verify}\colon \mathcal{PK} \times \mathcal{M} \times \mathcal{S} \to \{0,1\}$ *is the verification algorithm that takes a session public key* pk_{id}*, a message* m*, and a signature* σ *as input and outputs a bit* b*.*

By $\mathcal{PK}$*,* $\mathcal{SK}$*,* $\mathcal{ST}$*,* $\mathcal{ID}$*,* $\mathcal{M}$*, and* $\mathcal{S}$ *we denote the public key space, secret key space, wallet state space, identity space, message space, and signature space, respectively. For sake of simplicity, we assume that the key spaces for master keys and session keys are equal.*

Das et al. [DFL19] give a generic construction for a deterministic wallet from a signature scheme with rerandomizable keys and a random oracle. The construction is displayed in Figure 5.9.

Correctness of a deterministic wallet scheme requires that verification of a signature should succeed with overwhelming probability for any pair of matching session keys. It is formally defined below.

Definition 5.3.2 (Correctness of Stateful Wallets)**.** *Let* $\mathtt{SW}$ *be a deterministic wallet with algorithms* $(\mathtt{SW.KGen}, \mathtt{SW.RandSK}, \mathtt{SW.RandPK}, \mathtt{SW.Sign}, \mathtt{SW.Verify})$*. We define* $\mathtt{SW}$ *to be correct if for any* $n \in \mathbb{N}$*, any* $(mpk, msk, st_0) \leftarrow\!\!\$\ \mathtt{SW.KGen}(1^\lambda)$*, any* $i \in [n]$*, any* $id_1, \ldots, id_n \in \mathcal{ID}$*, and any* $m \in \mathcal{M}$ *it holds that*

$$\Pr[\mathtt{SW.Verify}(pk_i, m, \mathtt{SW.Sign}(sk_i, pk_i, m)) = 1] \geq 1 - \mathrm{negl}(\lambda)\,,$$

where

$$(sk_i, st_i) \leftarrow \mathtt{SW.RandSK}(msk, id_i, st_{i-1})\,,$$
$$(pk_i, st_i) \leftarrow \mathtt{SW.RandPK}(mpk, id_i, st_{i-1})\,.$$

As for security, again following [DFL19], deterministic wallets should provide both unlinkability and unforgeability. The former ensures that session public keys cannot be linked to one another, i.e., an adversary cannot tell whether two session public keys are derived from the same master public key. The latter extend the classical notion of unforgeability, i.e., no one without the secret key can produce valid signatures, to the setting of deterministic wallets. For both notions we give the formal definitions below.

$\underline{\texttt{SW.KGen}(1^\lambda)}$

$st \leftarrow\!\!\$\ \{0,1\}^\lambda$
$(mpk, msk) \leftarrow\!\!\$\ \texttt{RSig.KGen}(1^\lambda)$
return (mpk, msk, st)

$\underline{\texttt{SW.RandSK}(msk, id, st)}$

$(r, st) \leftarrow \mathsf{H}(st, id)$
$sk_{id} \leftarrow \texttt{RSig.RandSK}(msk, r)$
return (sk_{id}, st)

$\underline{\texttt{SW.Sign}(sk, pk, m)}$

$m' \leftarrow (m, pk)$
$\sigma \leftarrow\!\!\$\ \texttt{RSig.Sign}(sk, m')$
return σ

$\underline{\texttt{SW.RandPK}(mpk, id, st)}$

$(r, st) \leftarrow \mathsf{H}(st, id)$
$pk_{id} \leftarrow \texttt{RSig.RandPK}(mpk, r)$
return (pk_{id}, st)

$\underline{\texttt{SW.Verify}(pk, m, \sigma)}$

$m' \leftarrow (m, pk)$
return $\texttt{RSig.Verify}(pk, m', \sigma)$

Figure 5.9: Generic construction for a deterministic wallet from a signature scheme with rerandomizable keys RSig and a random oracle H.

Definition 5.3.3 (W-UNL Security)**.** *Let* SW *be a deterministic wallet and game* WUNL *be as defined in Figure 5.10. For any adversary $\mathcal{A}$, its corresponding* W-UNL *advantage is defined as*

$$\mathbf{Adv}^{\mathsf{WUNL}}_{\mathtt{SW}}(\mathcal{A}) \coloneqq \left|2\ \Pr[\mathsf{EUFCMAHRK}^{\mathcal{A}} \Rightarrow 1] - 1\right| .$$

We say that SW *is* W-UNL *secure if for any efficient adversary $\mathcal{A}$ it holds that*

$$\mathbf{Adv}^{\mathsf{WUNL}}_{\mathtt{SW}}(\mathcal{A}) \leq \mathrm{negl}(\lambda) .$$

Definition 5.3.4 (W-UNF Security)**.** *Let* SW *be a deterministic wallet and game* WUNF *be as defined in Figure 5.11. For any adversary $\mathcal{A}$, its corresponding* W-UNF *advantage is defined as*

$$\mathbf{Adv}^{\mathsf{WUNF}}_{\mathtt{SW}}(\mathcal{A}) \coloneqq \Pr[\mathsf{EUFCMAHRK}^{\mathcal{A}} \Rightarrow 1] .$$

We say that SW *is* W-UNF *secure if for any efficient adversary $\mathcal{A}$ it holds that*

$$\mathbf{Adv}^{\mathsf{WUNF}}_{\mathtt{SW}}(\mathcal{A}) \leq \mathrm{negl}(\lambda) .$$

5.3.2 Security

In this section we show that the generic construction achieves both unlinkability and unforgeability against quantum adversaries. Recall that since we are in the post-quantum setting, the oracles provided by the challenger in the unlinkability game (PK, getState,

Game WUNL

$b \leftarrow_{\$} \{0,1\}$
$(mpk, msk, st) \leftarrow \texttt{SW.KGen}(1^\lambda)$
$id^* \leftarrow \mathcal{A}_1^{\mathsf{WalSign},\mathsf{PK}}(mpk)$
if $(K[id^*] \neq \bot)$
  return 0
$(pk^0_{id^*}, st^*) \leftarrow \texttt{SW.RandPK}(mpk, id^*, st)$
$(sk^0_{id^*}, st^*) \leftarrow \texttt{SW.RandSK}(msk, id^*, st)$
$st \leftarrow st^*$
$(\overline{mpk}, \overline{msk}, st^*) \leftarrow \texttt{SW.KGen}(1^\lambda)$
$(pk^1_{id^*}, st^*) \leftarrow \texttt{SW.RandPK}(\overline{mpk}, id^*, st^*)$
$(sk^1_{id^*}, st^*) \leftarrow \texttt{SW.RandSK}(\overline{msk}, id^*, st^*)$
$K[id^*] \leftarrow (pk^b_{id^*}, sk^b_{id^*})$
$b' \leftarrow \mathcal{A}_2^{\mathsf{WalSign},\mathsf{PK},\mathsf{getState}}(mpk, pk^b_{id^*})$
return $(b' = b)$

oracle PK(id)

$(pk_{id}, st^*) \leftarrow \texttt{SW.RandPK}(mpk, id, st)$
$(sk_{id}, st^*) \leftarrow \texttt{SW.RandSK}(msk, id, st)$
$K[id] \leftarrow (pk_{id}, sk_{id})$
$st \leftarrow st^*$
return pk_{id}

oracle getState()

return st

oracle WalSign(m, id)

if $K[id] = \bot$
  return $\bot$
$(pk_{id}, sk_{id}) \leftarrow K[id]$
$\sigma \leftarrow \texttt{SW.Sign}(sk_{id}, m)$
return σ

Figure 5.10: Security game WUNL.

Game WUNF

$\mathcal{S} \leftarrow \emptyset$
$(mpk, msk, st) \leftarrow \texttt{SW.KGen}(1^\lambda)$
$(m^*, \sigma^*, id^*) \leftarrow \mathcal{A}^{\mathsf{WalSign},\mathsf{PK}}(mpk, st)$
if $(K[id^*] \neq \bot)$
  return 0
$(pk_{id^*}, sk_{id^*}) \leftarrow K[id^*]$
if $(m^*, id^*) \in \mathcal{S}$
return 0
if $\texttt{SW.Verify}(pk_{id^*}, m^*, \sigma^*) = 0$
  return 0
return 1

oracle PK(id)

$(pk_{id}, st^*) \leftarrow \texttt{SW.RandPK}(mpk, id, st)$
$(sk_{id}, st^*) \leftarrow \texttt{SW.RandSK}(msk, id, st)$
$K[id] \leftarrow (pk_{id}, sk_{id})$
$st \leftarrow st^*$
return pk_{id}

oracle WalSign(m, id)

if $K[id] = \bot$
  return $\bot$
$(pk_{id}, sk_{id}) \leftarrow K[id]$
$\sigma \leftarrow \texttt{SW.Sign}(sk_{id}, m)$
$\mathcal{S} \leftarrow_{\cup} (m, id)$
return σ

Figure 5.11: Security game WUNF.

WalSign) and in the unforgeability game (PK, WalSign) are executed on a classical computer. Hence, also the (quantum) adversary gets only classical access to these oracles. However, the adversary can use its quantum computing power to access the random oracle in superposition.

The following theorem shows the unlinkability. First, we provide a proof intuition of Theorem 5.3.5. Let us recall how the unlinkability property is proven in the ROM (cf. [DFL19]). Note that the wallet public keys are derived from the wallet state, which is stored within the wallet, hidden from the adversary. The classical adversary can then try to guess one of the states of the wallet and make a "problematic query" to the random oracle H on such a state, in order to derive one of the session public keys generated by the wallet. If the adversary guesses the wallet's state correctly, it can distinguish a public key generated by the wallet from a randomly generated one, hence, the adversary will be able to win the unlinkability game. The classical proof consists of two steps: (1) showing the probability that the adversary makes the problematic query to the random oracle, as mentioned above, is negligible, and (2) showing that the adversary has no advantage in winning the unlinkability game conditioned on the event that it does not make any problematic query. Finally, note that while this proof uses the stronger notion of rerandomizable public and secret keys (cf. Definition 2.2.14), it is easy to see that it also works with our relaxed definition of rerandomizable public keys (cf. Definition 2.2.16). This is because the unlinkability game requires the adversary to distinguish a public key generated by the wallet from a randomly generated public key.

Our proof in the QROM follows the same approach, however, the first step requires a different technique. Recall that the wallet state gets refreshed with every public key query. In [DFL19], the challenger keeps a list of the states of the wallet scheme—starting from the initial state till the one obtained during the last public key query. In the analysis a simple comparison allows to check whether a query by the classical adversary is problematic, i.e., whether it coincides with one of the states of the wallet. Since the adversary can access the random oracle H only classically (it can query on exactly one input at a time), hence the challenger can store all these queries in a list. In the QROM, however, we cannot keep such a list as the adversary now has quantum computation power, hence, the adversary can query the random oracle on several, and even all, inputs in superposition.[28] Instead, we consider a game hop which we can bound by the advantage of the adversary in distinguishing two random oracles, which in turn can be bound using the O2H lemma. For the resulting game, the same argument as in the classical proof applies, i.e., the rerandomizable property of the underlying signature schemes ensures that the adversary has no advantage in winning the game.

Theorem 5.3.5. *Let* RSig *be a signature scheme with rerandomizable public keys (cf. Definition 2.2.16) and* H *a random oracle. Let further* SW *be the stateful wallet scheme built from* RSig *and* H *(cf. Figure 5.9). Then, for any quantum adversary* $\mathcal{A}$ *it holds that*

$$\mathbf{Adv}^{\mathsf{WUNL}}_{\mathtt{SW}}(\mathcal{A}) \leq \mathrm{negl}(\lambda)\,.$$

[28] We note that, to some extent, the compressed oracle technique by Zhandry [Zha19a] allows the recording of superposition queries.

Proof. Throughout, let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be an adversary which makes q and q_{PK} queries to its oracles $|\mathsf{H}\rangle$ and PK, respectively. To prove the theorem we use the following two games.

Game G_0: This is game WUNL (cf. Figure 5.10) instantiated with $\mathtt{SW}$ (cf. Figure 5.9).

Game G_1: This is the same as G_0, except that the randomness r and the new state st^*, prior to running $\mathcal{A}_2$, are sampled at random, independent of the random oracle.

A standard reformulation yields

$$\begin{aligned}\mathbf{Adv}^{\mathsf{WUNL}}_{\mathtt{SW}}(\mathcal{A}) &= \mathbf{Adv}^{\mathsf{G}_0}(\mathcal{A}) \\ &= \left|2\Pr[\mathsf{G}_0^{\mathcal{A}} \Rightarrow 1] - 1\right| \\ &\leq 2\left|\Pr[\mathsf{G}_0^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathsf{G}_1^{\mathcal{A}} \Rightarrow 1]\right| + 2\Pr[\mathsf{G}_1^{\mathcal{A}} \Rightarrow 1] - 1 \\ &= 2\left|\Pr[\mathsf{G}_0^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathsf{G}_1^{\mathcal{A}} \Rightarrow 1]\right| + \mathbf{Adv}^{\mathsf{G}_1}(\mathcal{A})\,. \end{aligned} \tag{5.8}$$

In both games, the randomness and new state are distributed identical. The only difference lies in the random oracle. From point of view of $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, the random oracle in game G_1 is $|\mathsf{H}_{\mathcal{S}\mapsto\$}\rangle$, i.e., the random oracle that is reprogrammed to random values for every $x \in \mathcal{S}$, where $\mathcal{S}$ contains all pairs of states and IDs prior to running $\mathcal{A}_2$. Hence, we can bound the advantage in distinguishing G_0 and G_1 by the advantage in distinguishing the random oracles $|\mathsf{H}\rangle$ and $|\mathsf{H}_{\mathcal{S}\mapsto\$}\rangle$. Applying the O2H Lemma (cf. Lemma 2.6.1) yields

$$\begin{aligned}\left|\Pr[\mathsf{G}_0^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathsf{G}_1^{\mathcal{A}} \Rightarrow 1]\right| &= \left|\Pr[\mathcal{A}^{|\mathsf{H}\rangle} \Rightarrow 1] - \Pr[\mathcal{A}^{|\mathsf{H}_{\mathcal{S}\mapsto\$}\rangle} \Rightarrow 1]\right| \\ &\leq 2q\sqrt{\Pr[x \in \mathcal{S} \colon \mathcal{B}^{|\mathsf{H}\rangle} \Rightarrow x]}\,, \end{aligned} \tag{5.9}$$

where $\mathcal{B}$ is the adversary specified in Lemma 2.6.1. Observe that while $\mathcal{A}$ knows all identities in the set $\mathcal{S}$ (they are chosen by $\mathcal{A}$), $\mathcal{A}$ has no information about the states in set $\mathcal{S}$. By querying $\mathsf{getState}$, $\mathcal{A}$, more precisely, $\mathcal{A}_2$, learns the state, but only after the challenge public key was generated; by the one-wayness of the random oracle this does not yield any information about the states in $\mathcal{S}$. Furthermore, we have $|\mathcal{S}| \leq q_{\mathsf{PK}} + 2$ at any point in time, q_{PK} from $\mathcal{A}_1$'s queries and 2 queries from the challenge phase. This yields

$$\Pr[x \in \mathcal{S} \colon \mathcal{B}^{|\mathsf{H}\rangle} \Rightarrow x] \leq \frac{|\mathcal{S}|}{2^\lambda} \leq \frac{q_{\mathsf{PK}} + 2}{2^\lambda}\,. \tag{5.10}$$

Combining (5.9) and (5.10) yields

$$\left|\Pr[\mathsf{G}_0^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathsf{G}_1^{\mathcal{A}} \Rightarrow 1]\right| \leq \frac{q_{\mathsf{PK}} + 2}{2^\lambda} \leq \mathrm{negl}(\lambda)\,. \tag{5.11}$$

It remains to bound the advantage of $\mathcal{A}$ in game G_1, where the same argument from the classical proof applies. In G_1, the challenge public key $pk^b_{id^*}$ given to $\mathcal{A}_2$ is independent of the random oracle (as the random oracle is not used in G_1 any more for deriving keys). Hence, it is irrelevant whether the adversary makes any query (classical or quantum) to the random oracle. As $\mathtt{RSig}$ is a signature scheme with rerandomizable public keys, the challenge public keys $pk^0_{id^*}$ and $pk^1_{id^*}$ are identically distributed, which yields

$$\mathbf{Adv}^{\mathsf{G}_1}(\mathcal{A}) = 0\,. \tag{5.12}$$

Inserting (5.11) and (5.12) into (5.8) yields

$$\mathbf{Adv}^{\mathsf{WUNL}}_{\mathtt{SW}}(\mathcal{A}) \leq 2\,\mathrm{negl}(\lambda) + 0 \leq \mathrm{negl}(\lambda)\,.$$

This proves the theorem. □

The following theorem shows that the generic construction is unforgeable in the presence of quantum attackers. We briefly recap the classical proof in the ROM (cf. [DFL19]), thereby highlighting the challenge when switching to a quantum adversary. Note again, that the classical proof in the ROM uses the stronger notion of rerandomizable keys (cf. Definition 2.2.14) but also holds for the weaker notion of rerandomizable public keys (cf. Definition 2.2.16). The proof consists of a game hop in which the adversary loses the game if there is a collision of session keys for different identities. Due to the construction, this occurs if the random oracle outputs a collision which is bound by a simple counting argument. The advantage of an adversary in the resulting game is bound by the security of the underlying signature scheme using a reduction. The crucial part is that the reduction simulates the random oracle H for the adversary using its oracle Rand from the game $\mathsf{EUFCMAHRK}$. More precisely, for each query to H by the adversary, the reduction makes a query to Rand.

Our proof in the QROM follows the same idea, however additionally needs to take care of the use of the quantum random oracle $|\mathsf{H}\rangle$ by the adversary. The first part works exactly as in [DFL19], since the access to the oracle PK remains classical even for a quantum adversary. The second part, however, does not work as in [DFL19]. While the adversary can query the quantum random oracle $|\mathsf{H}\rangle$ in superposition, the reduction can query its oracle Rand only classically as it is provided by its (classical) challenger. By querying $|\mathsf{H}\rangle$ on an equal superposition of all (i.e., exponentially many) inputs, the reduction would need exponentially many queries to Rand in order to simulate $|\mathsf{H}\rangle$ for the adversary. Clearly, this would render the reduction useless as it would not be efficient. To tackle this issue, we do an additional game hop in which we switch from a random oracle to an oracle drawn from a small-range distribution. While this affects the advantage of the adversary only negligibly, it allows us to construct a reduction which can simulate the quantum random oracle for the adversary by making, a priori to running the adversary, a polynomial number of (classical) queries to its oracle Rand.

Theorem 5.3.6. *Let* $\mathtt{RSig}$ *be a signature scheme with rerandomizable public keys (cf. Definition 2.2.16) and* H *a random oracle. Let further* $\mathtt{SW}$ *be the stateful wallet scheme built from* $\mathtt{RSig}$ *and* H *(cf. Figure 5.9). Assume that* $\mathtt{RSig}$ *is* EUF-CMA-HRK *secure. Then, for any quantum adversary* $\mathcal{A}$ *it holds that*

$$\mathbf{Adv}^{\mathsf{WUNF}}_{\mathtt{SW}}(\mathcal{A}) \leq \mathrm{negl}(\lambda)\,.$$

Proof. Let G_0 be game WUNF (cf. Figure 5.11) instantiated with $\mathtt{SW}$ (cf. Figure 5.9). For sake of contradiction, we assume that there is an adversary $\mathcal{A}$ that has non-negligible advantage ϵ. More precisely, there exists a polynomial $p = p(\lambda)$ such that $p(\lambda) > \frac{1}{\epsilon(\lambda)}$ and

$$\mathbf{Adv}^{\mathsf{G}_0}(\mathcal{A}) = \epsilon\,.$$

Let G_1 be the slightly different game in which the adversary loses if there occurs a collision in the session keys, i.e., there exist $id_0 \neq id_1$ such that $K[id_0] = K[id_1]$. To detect the difference between the games such a collision has to occur. By construction and the unique key property of $\mathtt{RSig}$, this only happens if there is a collision in the random oracle outputs during queries to PK. Since $\mathcal{A}$ can access PK only classically, the argument from the classical proof [DFL19] applies which states that such a collision occurs only with negligible probability. We get

$$\mathbf{Adv}^{\mathsf{G}_1}(\mathcal{A}) = \epsilon - \mathrm{negl}(\lambda)\,.$$

Now consider game G_2, where the random oracle H is replaced by an oracle where the first output (the randomness) is drawn from a small-range distribution using $l = 2Cq^3p$ samples. Here, C is the constant from Lemma 2.6.3, q is the number of random oracle queries by $\mathcal{A}$, and p is the polynomial defined above. Lemma 2.6.3 states that the adversary can detect this change with probability at most $\frac{Cq^3}{l} = \frac{Cq^3}{2Cq^3p} = \frac{1}{2p}$. This yields

$$\mathbf{Adv}^{\mathsf{G}_2}(\mathcal{A}) = \epsilon - \mathrm{negl}(\lambda) - \frac{1}{2p}\,.$$

In the following we construct an adversary $\mathcal{B}$ playing game $\mathsf{EUFCMAHRK}$ against $\mathtt{RSig}$ using adversary $\mathcal{A}$ playing G_2. For simplicity and without loss of generality we assume that $\mathcal{A}$ never makes a query resulting in $\perp$ and that there are no collisions for the session keys. Upon receiving a public key pk, it makes $l = 2Cq^3p$ queries to Rand, samples an initial sate $st_0 \leftarrow\!\!\$\ \{0,1\}^\lambda$, and runs $\mathcal{A}$ on input $(mpk = pk, st_0)$. In order to simulate the random oracle, $\mathcal{B}$ uses a small-range function for the first output (the randomness) using the l samples obtained from the queries to Rand prior to running $\mathcal{A}$; for the second output (the wallet state), $\mathcal{B}$ uses a $2q$-wise independent function. When $\mathcal{A}$ makes a query id to PK, $\mathcal{B}$ computes (r_{id}, st^*), $pk_{id} \leftarrow \mathtt{RSig.RandPK}(pk; r_{id})$, sets $K[id] \leftarrow (pk_{id}, r_{id})$, and sends pk_{id} to $\mathcal{A}$. Queries (m, id) to $\mathsf{WalSign}$ are processed by first obtaining $(pk_{id}, r_{id}) = K[id]$ and then computing $m' \leftarrow (m, pk_{id})$. Subsequently, $\mathcal{B}$ queries Sign on (m', r_{id}) to obtain σ which it forwards to $\mathcal{A}$. Eventually, $\mathcal{A}$ outputs a forgery (m^*, σ^*, id^*); $\mathcal{B}$ obtains $(pk_{id^*}, r_{id^*}) = K[id^*]$ and outputs $((m^*, pk_{id^*}), \sigma^*, r_{id^*})$ as its forgery. It follows that $\mathcal{B}$ perfectly simulates game G_2 for $\mathcal{A}$ and it remains to argue that $\mathcal{B}$ wins whenever $\mathcal{A}$ wins. First, given that (m^*, σ^*, id^*) is a valid forgery, it holds that $\mathcal{A}$ never queried (m^*, id^*) to $\mathsf{WalSign}$ and, by construction, $\mathcal{B}$ never queried $((m^*, pk_{id^*}), r_{id^*})$ to its oracle Sign. Second, it holds that $r_{id^*} \in \mathcal{T}$ (in game $\mathsf{EUFCMAHRK}$). This follows trivially as $\mathcal{A}$'s random oracle is simulated using a small-range distribution using l samples which $\mathcal{B}$ obtains from the initial l queries to Rand. Third, validity of the forged signature by $\mathcal{A}$ implies validity of the forged signature by $\mathcal{B}$, precisely

$$\mathtt{RSig.Verify}(pk_{id^*}, (m^*, pk_{id^*}), \sigma^*) = \mathtt{SW.Verify}(pk_{id^*}, m^*, \sigma^*) = 1\,.$$

Assuming $\mathtt{RSig}$ to be EUF-CMA-HRK secure therefore yields

$$\mathbf{Adv}^{\mathsf{G}_2}(\mathcal{A}) \leq \mathrm{negl}(\lambda)\,.$$

Now observe that

$$\frac{1}{p} = \frac{1}{p} - \frac{1}{2p} \leq \epsilon - \frac{1}{2p} \leq \mathrm{negl}(\lambda)\,,$$

which yields a contradiction to the initial assumption that ϵ is non-negligible. Thus ϵ is indeed negligible and it holds that

$$\mathbf{Adv}^{\mathsf{WUNF}}_{\mathtt{SW}}(\mathcal{A}) = \mathbf{Adv}^{\mathsf{G}_0}(\mathcal{A}) = \epsilon \leq \mathrm{negl}(\lambda)\,.$$

This proves the theorem. □

5.4 Post-Quantum Security of Yao's Garbled Circuits

Another important research area is multi-party computation (MPC) [Yao86, GMW87]. Here, two or more parties jointly evaluate a function on their respective inputs without revealing anything about the inputs to the other party. However, there is not much research done considering the post-quantum security of MPC.

In this section, we prove that Yao's garbled circuits protocol (cf. Section 5.4.1) achieves post-quantum security if each of the underlying building blocks is replaced with a post-quantum secure variant. As this seems intuitive, recall that a simple switch to post-quantum secure building blocks is not always sufficient [Gag17]. An example for this is the Fiat-Shamir transformation. Simply constructing a signature scheme based on a quantum hard problem is not sufficient to achieve security in the QROM.

The classical security proof of Yao's protocol is due to Lindell and Pinkas [LP09]. They showed that a secure OT protocol and a symmetric key encryption scheme, which is secure under double encryption (a security notion they introduced), are sufficient to prove Yao's protocol secure against semi-honest adversaries. Concerning the security under double encryption, they show that, classically, IND-CPA security implies security under double encryption. We show that both proofs can be lifted to quantum adversaries. Regarding the proof for the protocol, this is relatively straightforward, by arguing about the different steps from the classical proof. As for the security under double encryption, we directly prove the post-quantum security since the classical proof is merely sketched. Furthermore, we conduct the proof in the QROM, whereas the classical proof sketch does not consider random oracles. This is relevant when one wants to use encryption schemes where the proofs are naturally in the QROM. An obvious example is the sponge-based encryption scheme SLEnc underlying Slae, developed in Chapter 3.

5.4.1 Description of Yao's Protocol

Yao's garbled circuits protocol [Yao82] is a fundamental secure two-party computation protocol. The protocol consists of two cryptographic primitives: a symmetric key encryption scheme and an OT protocol. It is executed by two parties, the garbler $\mathcal{G}$ and the evaluator $\mathcal{E}$ with corresponding inputs x and y. At the end of the protocol, both parties want to obtain $\mathcal{F}(x, y)$ for a deterministic function $\mathcal{F}$. At the start of the protocol, both parties agree on a Boolean circuit that evaluates $\mathcal{F}$.

The garbler $\mathcal{G}$ starts by choosing two keys k_i^0 and k_i^1 for each wire w_i in the circuit, which represent the possible values 0 and 1. For a gate g_j, let l, r, and o denote the indices of the left input wire, right input wire, and output wire, respectively. Let $k_o^{g_j(x,y)}$ denote the output key for gate j corresponding to the plaintext inputs x and y. Then $\mathcal{G}$ generates the garbled table

$$c_0 \leftarrow \mathtt{Enc}\left(k_l^0, \mathtt{Enc}\left(k_r^0, k_o^{g_j(0,0)}\right)\right) \qquad c_1 \leftarrow \mathtt{Enc}\left(k_l^0, \mathtt{Enc}\left(k_r^1, k_o^{g_j(0,1)}\right)\right)$$
$$c_2 \leftarrow \mathtt{Enc}\left(k_l^1, \mathtt{Enc}\left(k_r^0, k_o^{g_j(1,0)}\right)\right) \qquad c_3 \leftarrow \mathtt{Enc}\left(k_l^1, \mathtt{Enc}\left(k_r^1, k_o^{g_j(1,1)}\right)\right)$$

for each gate g_j in the circuit. Following this, $\mathcal{G}$ sends the garbled tables (permuted using a secret random permutation), called the garbled circuit $G(C)$, along with the keys corresponding to its input x to $\mathcal{E}$. That is, if its input bit on wire w_i is 1 it sends k_i^1, otherwise, it sends k_i^0. Next, $\mathcal{E}$ obliviously receives the keys corresponding to its inputs from $\mathcal{G}$ by executing an OT protocol. For every gate g_j, $\mathcal{E}$ knows two out of the four input keys, which allows to decrypt exactly one entry of the garbled table and yields the corresponding output key. After evaluating the circuit, $\mathcal{E}$ obtains the keys assigned to the labels of the output wires of the circuit. In the final step, $\mathcal{G}$ sends over a mapping from the circuit output keys to the actual bit values and $\mathcal{E}$ shares the result with $\mathcal{G}$.

In the description, it is required that $\mathcal{E}$ can decrypt exactly one entry from the garbled table per gate, which is ensured by the properties elusive and efficiently verifiable range, defined below, followed by the correctness of Yao's GC protocol.

Definition 5.4.1 (Elusive and Efficiently Verifiable Range [LP09])**.** *Let* Ske *be a symmetric key encryption scheme with algorithms* $(\mathtt{Enc}, \mathtt{Dec})$ *and define the range of a key as* $\mathsf{Range}_\lambda(k) = \{\mathtt{Enc}(k,m)\}_{m\in\{0,1\}^\lambda}$.

1. *We say that* Ske *has an* elusive range*, if for any algorithm* $\mathcal{A}$ *it holds that*

$$\Pr[c \in \mathsf{Range}_\lambda(k) \,|\, \mathcal{A}(1^n) \Rightarrow c] \leq \mathrm{negl}(\lambda)\,,$$

   where the probability is taken over the keys.

2. *We say that* Ske *has an* efficiently verifiable range*, if there exists a probabilistic polynomial time machine* M *such that* $M(k,c) \Rightarrow 1$ *if and only if* $c \in \mathsf{Range}_\lambda(k)$.

Theorem 5.4.2 (Correctness of Yao's GC Protocol [LP09])**.** *We assume without loss of generality that* $x = x_1 \parallel \cdots \parallel x_n$ *and* $y = y_1 \parallel \cdots \parallel y_n$ *are two* n*-bit inputs for a Boolean circuit* C*. Let* $k_1, \ldots, k_n$ *be the labels of the circuit-input wires corresponding to* x*, and* $k_{n+1}, \ldots, k_{2n}$ *the labels of the circuit-input wires corresponding to* y*. Assume that the encryption scheme used to construct the garbled circuit* $G(C)$ *has an elusive and efficiently verifiable range. Then, given* $G(C)$*, and the strings* $k_1^{x_1}, \ldots, k_n^{x_n}, k_{n+1}^{y_1}, \ldots, k_{2n}^{y_n}$*, it is possible to compute* $C(x,y)$*, except with negligible probability.*

5.4.2 Protocol Security

In this section, we prove that Yao's protocol is post-quantum secure against semi-honest quantum adversaries. Recall that in this setting, the adversary can perform local quantum computations and tries to obtain additional information while genuinely running the protocol.

The restriction to local quantum computations is due to the post-quantum setting, in which only the adversary has quantum power while all other parties, in this case the protocol partner, remain classical. By restricting the adversary to be semi-honest, we ensure that it does not deviate from the protocol specification. This models a typical scenario of an adversary which tries to obtain additional information without being noticed by the other party. One can think of a computer virus affecting one of the protocol participants, which tries to remain unnoticed.

The theorem below states the post-quantum security of Yao's GC protocol given that both the OT protocol and the encryption scheme are post-quantum secure. Unlike the other proofs in this thesis which are game-based, the proof by Lindell and Pinkas is simulation-based [Lin17]. The idea is as follows. A scheme is deemed secure if there exists a simulator—hence the name simulation-based—which can simulate the view (what they see as part of the protocol) of the adversary. The crucial part is that the simulator only gets the input and output of the protocol. In context of Yao's GC there is a distinction whether the garbler or the evaluator is corrupted as both have a different view; for the garbler it is the output of the different OT protocols and the final output of the circuit, for the evaluator it is the garbled circuit, the keys corresponding to the garbler's input and the output of the different OT protocols.

Theorem 5.4.3 (Post-Quantum Security of Yao's GC Protocol)**.** *Let $\mathcal{F}$ be a deterministic function. Suppose that the oblivious transfer protocol is post-quantum secure against semi-honest adversaries, the encryption scheme is* pq-2Enc*-secure*[29]*, and the encryption scheme has an elusive and efficiently verifiable range. Then, the protocol described in Section 5.4.1 securely computes $\mathcal{F}$ in the presence of semi-honest quantum adversaries.*

Proof. We show that the classical proof by Lindell and Pinkas [LP09] carries over to the post-quantum setting. The proof is divided into two cases, one for a semi-honest garbler and one for a semi-honest evaluator. For each of these, the classical proof constructs a simulator which simulates the view of the corrupted party solely based on its input and output. Then the proof is based on a hybrid argument showing that the simulated view and the real view are computationally indistinguishable. Note that the real view remains the same when switching to the post-quantum setting, i.e., a quantum adversary. This is due to the fact that the adversary runs the protocol with an honest classical party. The mere difference is that the adversary can perform local quantum computation in order to break the security of the protocol.

Garbler corrupted. The view of the garbler $\mathcal{G}$ consists of its view from the OTs and the message (the circuit output) it obtains from the evaluator $\mathcal{E}$ at the end of the protocol.

Simulator $\mathcal{S}_{\mathcal{G}}$ for a corrupted garbler receives as input $(x, \mathcal{F}(x, y))$ and outputs the view $(x, r, \mathcal{S}_{\mathcal{G}}^{OT}(k_{n+1}^0, k_{n+1}^1), \ldots, \mathcal{S}_{\mathcal{G}}^{OT}(k_{2n}^0, k_{2n}^1), \mathcal{F}(x, y))$, where r is a random tape that $\mathcal{S}_{\mathcal{G}}$ uses to generate the garbled circuit and $\mathcal{S}_{\mathcal{G}}^{OT}(k_{n+i}^0, k_{n+i}^1)$ is the simulator for the i-th OT execution, which exists by the post-quantum security of the OT protocol.

To prove that the simulated view is computationally indistinguishable from the real view, the classical proof uses a sequence of hybrid distributions $H_0, \ldots, H_n$. In hybrid H_i,

[29] We formally define post-quantum security under double encryption (pq-2Enc security) in Definition 5.4.4.

the first i OT executions are real while the remaining $n - i$ OT executions are simulated. Lindell and Pinkas show that a distinguisher $\mathcal{D}$ for any two consecutive hybrids H_{i-1} and H_i can be transformed into an adversary $\mathcal{A}$ that distinguishes between the real view of the OT protocol and the simulated one. In our case, $\mathcal{D}$ is a quantum algorithm which entails that $\mathcal{A}$ is a quantum as well. Given that the OT protocol is post-quantum secure, we conclude that the hybrids H_0 and H_n are indistinguishable as every consecutive hybrid H_{i-1} and H_i are indistinguishable. Note that H_n is not yet the real view of the protocol, since the simulator uses $\mathcal{F}(x, y)$ to simulate the message that the evaluator sends to the garbler at the end of the protocol. More precisely, the simulator always sends the correct result, while the real protocol execution might result in the garbler sending an incorrect result. The indistinguishability between the real view and H_n follows from the correctness of the protocol (cf. Theorem 5.4.2), which states the garbler will not send $\mathcal{F}(x, y)$ only with negligible probability. This holds regardless of the setting and therefore also in the post-quantum setting. Based on this, we conclude that the real view is indistinguishable from the output of $\mathcal{S}_\mathcal{G}$.

Evaluator corrupted. The view of the evaluator $\mathcal{E}$ consists of a garbled circuit and n keys, which it receives from the garbler, as well as its view from the OTs.

The simulator $\mathcal{S}_\mathcal{E}$ for a corrupted evaluator receives as input $(y, \mathcal{F}(x, y))$ and outputs $(y, \widetilde{G}(C), k_1, \ldots, k_n, \mathcal{S}_\mathcal{E}^{OT}(y_1, k_{n+1}), \ldots, \mathcal{S}_\mathcal{E}^{OT}(y_n, k_{2n}))$, where $\widetilde{G}(C)$ is the 'fake' garbled circuit which evaluates to $\mathcal{F}(x, y)$ irrespectively of the used keys, $k_1, \ldots, k_n$ are randomly sampled keys, and $\mathcal{S}_\mathcal{E}^{OT}(y_i, k_{n+i})$ is the simulator for the i-th OT execution, which exists by the post-quantum security of the OT protocol.

The same argument as above yields that the output of $\mathcal{S}_\mathcal{E}$ is computationally indistinguishable from H_0, where the OT executions are replaced with the real view, since the OT protocol is post-quantum secure. The difference between the real view of $\mathcal{E}$ and H_0 is that the latter contains the 'fake' garbled circuit $\widetilde{G}(C)$. To show these are computationally indistinguishable, another sequence of hybrids $H_0, \ldots, H_{|C|}$ is introduced, where the first i gates[30] correspond to the 'fake' garbled circuit $\widetilde{G}(C)$, while the remaining $|C| - i$ gates correspond to the real garbled circuit $G(C)$. Lindell and Pinkas show that any distinguisher $\mathcal{D}$ between hybrids H_{i-1} and H_i can be transformed into an adversary $\mathcal{A}$ against the security of the encryption scheme. Assuming the encryption scheme to be pq-2Enc secure then yields that the hybrid can be distinguished only with negligible advantage.

Combined with the result above, this concludes the proof. □

5.4.3 Security against Double Encryption

To securely instantiate Yao's protocol, an encryption scheme which is secure under double encryption is required. In the classical setting, Lindell and Pinkas [LP09] provide a short sketch that the standard security notion for encryption schemes (IND-CPA) implies security under double encryption. In this section, we show that the same argument holds in the post-quantum setting, i.e., pqIND-CPA security implies post-quantum security under

[30] Using a topological ordering for the gates in the circuit.

double encryption (pq-2Enc). Furthermore, we extend the result to the QROM. This allows to cover a wider class of encryption schemes compared to the proof sketch from [LP09] which does not consider random oracles.

We start by introducing the post-quantum variant of the double encryption security game in the QROM (cf. Figure 5.12). Similar to the pqINDCPA game (cf. Figure 5.1), the adversary has to distinguish between the encryption of messages of its choice. The main difference is that there are four secret keys involved in the game, from which two are given to the adversary. As challenge messages, the adversary provides three pairs of messages. For each pair, one message is encrypted twice using two different keys from which at least one is unknown to the adversary. The adversary wins the game if it can distinguish which messages have been encrypted. It is granted access to two learning oracles which encrypt messages under a combination of a key given by the adversary and one of the unknown keys. There are two differences between our notion and the (classical) one given in [LP09]. First, we allow for multiple challenge queries from the adversary while [LP09] allows merely one. Second, the two known keys are honestly generated by the challenger and then handed over to the adversary. In [LP09], the adversary chooses these keys by itself. Since these keys correspond to the keys that the garbler generates honestly and obliviously sends to the evaluator, this change in the security notion models the actual scenario very well. In fact, the proof of Yao's protocol only requires the adversary to know two of the keys but not being able to generate them at will.

Definition 5.4.4 (Post-Quantum Security under Double Encryption)**.** *Let* SKE *be a symmetric key encryption scheme with algorithms* $(\mathtt{Enc}, \mathtt{Dec})$ *and let the game* pq2Enc *be defined as in Figure 5.12. Then, for any quantum adversary* $\mathcal{A}$ *its advantage against the double encryption security is defined as*

$$\mathbf{Adv}^{\mathsf{pq2Enc}}_{\mathrm{SKE}}(\mathcal{A}) \coloneqq \left|2\Pr[\mathsf{pq2Enc}^{\mathcal{A}} \Rightarrow 1] - 1\right| .$$

We say that SKE *is* pq-2Enc*-secure if* $\mathbf{Adv}^{\mathsf{pq2Enc}}_{\mathrm{SKE}}(\mathcal{A})$ *is negligible.*

Game pq2Enc

$b \leftarrow_{\$} \{0,1\}$
$k_0, k_1, k'_0, k'_1 \leftarrow_{\$} \mathcal{K}$
$b' \leftarrow \mathcal{A}^{\mathsf{Enc},\overline{\mathsf{Enc}_0},\overline{\mathsf{Enc}_1},\mathsf{H}}(k_0, k_1)$
return $(b' = b)$

oracle $\overline{\mathsf{Enc}}(m_0, m_1)$

parse m_0 **as** $x_0 \parallel y_0 \parallel z_0$
parse m_1 **as** $x_1 \parallel y_1 \parallel z_1$
$c_1 \leftarrow \mathtt{Enc}(k_0, \mathtt{Enc}(k'_1, x_b))$
$c_2 \leftarrow \mathtt{Enc}(k'_0, \mathtt{Enc}(k_1, y_b))$
$c_3 \leftarrow \mathtt{Enc}(k'_0, \mathtt{Enc}(k'_1, z_b))$
$c \leftarrow (c_1, c_2, c_3)$
return c

oracle $\overline{\mathsf{Enc}_0}(k, m)$

$c \leftarrow \mathtt{Enc}(k'_0, \mathtt{Enc}(k, m))$
return c

oracle $\overline{\mathsf{Enc}_1}(k, m)$

$c \leftarrow \mathtt{Enc}(k, \mathtt{Enc}(k'_1, m))$
return c

oracle $\mathsf{H}(\sum \alpha_{x,y} |x, y\rangle)$

return $\sum \alpha_{x,y} |x, y \oplus \mathsf{H}(x)\rangle$

Figure 5.12: Game pq2Enc to define post-quantum security under double encryption.

The theorem below states that pqIND-CPA security implies pq-2Enc security.

Theorem 5.4.5. *Let* SKE = (Enc, Dec) *be a symmetric key encryption scheme. Then, for any quantum adversary* $\mathcal{A}$ *against the post-quantum security under double encryption security of* SKE*, there exists a quantum adversary* $\overline{\mathcal{A}}$ *against the* pqIND-CPA *security of* SKE *such that*

$$\mathbf{Adv}^{\mathsf{pq2Enc}}_{\mathrm{SKE}}(\mathcal{A}) \leq 3\,\mathbf{Adv}^{\mathsf{pqINDCPA}}_{\mathrm{SKE}}(\overline{\mathcal{A}})\,.$$

Proof. Intuitively, the game pq2Enc (cf. Figure 5.12) looks very much akin to pqINDCPA, except that more keys are involved and each challenge query consists of several message pairs. However, simply reducing the pqIND-CPA advantage of SKE to the winning game pq2Enc does not work. The issue is that every message part is encrypted using a different combination of keys. Consider the following reduction which uses the game pqINDCPA to simulate encryptions under key k_1'. For every challenge $(x_0, y_0, z_0), (x_1, y_1, z_1)$, the adversary forwards (x_0, x_1) and (z_0, z_1) to its challenge oracle from the pqINDCPA game and encrypts the result using k_0 and k_0' which it samples by itself. In order to simulate the correct challenge oracle for the adversary, it has to encrypt y_b under keys k_1 and k_0'. While the reduction knows both keys, it is unaware of the message it has to encrypt, as this would trivially allow to win the pqINDCPA game.

To circumvent this issue, we introduce another game G (cf. Figure 5.13). In this game, the second ciphertext part c_2 always contains the encryption of y_0, irrespectively of the secret bit b. By removing the issue described above, this allows to transform any adversary against G into an adversary against pqINDCPA. It remains to bound the advantage in distinguishing between pq2Enc and G, which, in turn, can also be bound by the pqIND-CPA security of SKE.

Let $\mathcal{A}$ be a quantum adversary against pq2Enc. A simple reformulation yields

$$\begin{aligned}
\mathbf{Adv}^{\mathsf{pq2Enc}}(\mathcal{A}) &= \left|2\,\Pr[\mathsf{pq2Enc}^{\mathcal{A}} \Rightarrow 1] - 1\right| \\
&= \left|2\,\left(\Pr[\mathsf{pq2Enc}^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathsf{G}^{\mathcal{A}} \Rightarrow 1] + \Pr[\mathsf{G}^{\mathcal{A}} \Rightarrow 1]\right) - 1\right| \\
&\leq 2\,\left|\Pr[\mathsf{pq2Enc}^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathsf{G}^{\mathcal{A}} \Rightarrow 1]\right| + \mathbf{Adv}^{\mathsf{G}}(\mathcal{A})\,. \qquad (5.13)
\end{aligned}$$

We start by bounding the advantage in distinguishing pq2Enc and G. The games are identical if $b = 0$ which leads to

$$\left|\Pr[\mathsf{pq2Enc}^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathsf{G}^{\mathcal{A}} \Rightarrow 1]\right| = \left|\Pr[\mathcal{A}^{\mathsf{pq2Enc}} \Rightarrow 1 \,|\, b = 1] - \Pr[\mathcal{A}^{\mathsf{G}} \Rightarrow 1 \,|\, b = 1]\right|\,.$$

Now we construct a quantum adversary $\mathcal{B}_1$ playing the pqINDCPA game using $\mathcal{A}$ as a subroutine, where the symmetric key k from game pqINDCPA corresponds to k_0' in pq2Enc and G. It samples three keys k_0, k_1, and k_1', and sends the former two to the adversary $\mathcal{A}$. For every query (k, m) that $\mathcal{A}$ makes to $\overline{\mathsf{Enc}_1}$, $\mathcal{B}_1$ answers with the ciphertext obtained by first encrypting m using key k_1' and then encrypting the result using key k. For queries (k, m) to $\overline{\mathsf{Enc}_0}$, $\mathcal{B}_1$ locally computes $\mathtt{Enc}(k, m)$, invokes its own oracle Enc on it, and forwards the response to $\mathcal{A}$. For challenge queries (m_0, m_1), $\mathcal{B}_1$ first parses m_0 as $x_0 \parallel y_0 \parallel z_0$ and m_1 as $x_1 \parallel y_1 \parallel z_1$. The ciphertext c_1 is computed locally by encrypting x_1 using keys k_1' and k_0. For the ciphertext c_3, $\mathcal{B}_1$ encrypts z_1 using k_1', queries its own oracle Enc on the result, and sets c_3 to the response. As for c_2, $\mathcal{B}_1$ encrypts both y_0 and y_1

using k_1, invokes its own challenge oracle LR-Enc on the two ciphertexts, and assigns the response to c_2. For each of the above queries, $\mathcal{B}_1$ invokes its own quantum random oracle whenever the local simulation of the encryption algorithm requires it. Queries $\sum \alpha_{x,y} |x, y\rangle$ to the quantum random oracle by $\mathcal{A}$ are answered by forwarding the query to the quantum random oracle from game pqINDCPA and sending the response $\sum \alpha_{x,y} |x, y \oplus \mathsf{H}(x)\rangle$ back to $\mathcal{A}$. When $\mathcal{A}$ outputs a bit b', $\mathcal{B}_1$ simply forwards b' as its own output.

It is easy to see that $\mathcal{B}_1$ perfectly simulates pq2Enc and G conditioned on $b = 1$ and $b = 0$, respectively, where b is the secret bit from game pqINDCPA. This yields

$$\begin{aligned}
\left|\Pr[\mathsf{pq2Enc}^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathsf{G}^{\mathcal{A}} \Rightarrow 1]\right| &= \left|\Pr[\mathcal{A}^{\mathsf{pq2Enc}} \Rightarrow 1 \,|\, b = 1] - \Pr[\mathcal{A}^{\mathsf{G}} \Rightarrow 1 \,|\, b = 1]\right| \\
&= \Big|\Pr[\mathcal{B}_1^{\mathsf{pqINDCPA}} \Rightarrow 1 \,|\, b = 1] \\
&\quad - \Pr[\mathcal{B}_1^{\mathsf{pqINDCPA}} \Rightarrow 1 \,|\, b = 0]\Big| \\
&= \mathbf{Adv}^{\mathsf{pqINDCPA}}(\mathcal{B}_1)\,. \qquad (5.14)
\end{aligned}$$

Next, we show that any quantum adversary $\mathcal{A}$ against G can be transformed into an adversary $\mathcal{B}_2$ against pqINDCPA. The idea is as follows: $\mathcal{B}_2$ samples all keys but k_1' by itself. Encryptions using key k_1' are simulated using its own oracles Enc and LR-Enc from game pqINDCPA while all other encryptions are performed locally.

At the start of the game, $\mathcal{B}_2$ chooses three keys k_0, k_1, and k_0' at random and sends k_0 and k_1 to $\mathcal{A}$. Queries (k, m) to $\overline{\mathsf{Enc_0}}$ can be simulated using only the quantum random oracle, as $\mathcal{B}_2$ knows both k and k_0'. Queries (k, m) to $\overline{\mathsf{Enc_1}}$ require to first invoke the oracle Enc on m before encrypting the response using key k, again querying the quantum random oracle as required by the encryption algorithm. For the challenge queries (m_0, m_1), $\mathcal{B}_2$ first parses m_0 and m_1 as $x_0 \parallel y_0 \parallel z_0$ and $x_1 \parallel y_1 \parallel z_1$, respectively. Recall that in G the value y_1 is never encrypted. Hence, the second ciphertext c_2 is computed locally, only invoking the quantum random oracle, by encrypting y_0 using the keys k_0' and k_1. As for the other challenge messages (x_0, x_1, z_0, z_1), $\mathcal{B}_2$ invokes its own challenge oracle LR-Enc twice, namely on (x_0, x_1) and (z_0, z_1). The first response is encrypted using key k_0 yielding c_1, while the second response is encrypted using key k_0' yielding c_3. Finally, the ciphertexts are sent back to $\mathcal{A}$. Just as in the first part, queries $\sum \alpha_{x,y} |x, y\rangle$ that $\mathcal{A}$ makes to its quantum random oracle are forwarded to the quantum random oracle from the pqINDCPA game, as is the response $\sum \alpha_{x,y} |x, y \oplus \mathsf{H}(x)\rangle$. Whenever $\mathcal{A}$ outputs a bit b', $\mathcal{B}_2$ simply outputs the same bit.

It is easy to see that $\mathcal{B}_2$ perfectly simulates G for $\mathcal{A}$, where the secret key from game pqINDCPA corresponds to the key k_1' in G. By forwarding the output of $\mathcal{A}$, we end up with

$$\begin{aligned}
\mathbf{Adv}^{\mathsf{G}}(\mathcal{A}) &= \left|\Pr[\mathcal{A}^{\mathsf{G}} \Rightarrow 1 \,|\, b = 1] - \Pr[\mathcal{A}^{\mathsf{G}} \Rightarrow 1 \,|\, b = 0]\right| \\
&= \left|\Pr[\mathcal{B}_2^{\mathsf{pqINDCPA}} \Rightarrow 1 \,|\, b = 1] - \Pr[\mathcal{B}_2^{\mathsf{pqINDCPA}} \Rightarrow 1 \,|\, b = 0]\right| \\
&= \mathbf{Adv}^{\mathsf{pqINDCPA}}(\mathcal{B}_2)\,. \qquad (5.15)
\end{aligned}$$

Inserting (5.14) and (5.15) into (5.13); defining $\overline{\mathcal{A}}$ to be the adversary with the higher

advantage among $\mathcal{B}_1$ and $\mathcal{B}_2$, we finally obtain

$$\begin{aligned}\mathbf{Adv}^{\mathsf{pq2Enc}}(\mathcal{A}) &\leq 2\left|\Pr[\mathsf{pq2Enc}^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathsf{G}^{\mathcal{A}} \Rightarrow 1]\right| + \mathbf{Adv}^{\mathsf{G}}(\mathcal{A}) \\ &\leq 2\,\mathbf{Adv}^{\mathsf{pqINDCPA}}(\mathcal{B}_1) + \mathbf{Adv}^{\mathsf{pqINDCPA}}(\mathcal{B}_2) \\ &\leq 3\,\mathbf{Adv}^{\mathsf{pqINDCPA}}(\overline{\mathcal{A}})\,.\end{aligned}$$

In particular, pqIND-CPA security implies pq-2Enc security. □

Game G	**oracle** $\overline{\mathsf{Enc}}(m_0, m_1)$	**oracle** $\overline{\mathsf{Enc_0}}(k, m)$
$b \leftarrow_{\$} \{0,1\}$	**parse** m_0 **as** $x_0 \parallel y_0 \parallel z_0$	$c \leftarrow \mathtt{Enc}(k'_0, \mathtt{Enc}(k, m))$
$k_0, k_1, k'_0, k'_1 \leftarrow_{\$} \mathcal{K}$	**parse** m_1 **as** $x_1 \parallel y_1 \parallel z_1$	**return** c
$b' \leftarrow \mathcal{A}^{\overline{\mathsf{Enc}}, \overline{\mathsf{Enc_0}}, \overline{\mathsf{Enc_1}}, \mathsf{H}}(k_0, k_1)$	$c_1 \leftarrow \mathtt{Enc}(k_0, \mathtt{Enc}(k'_1, x_b))$	**oracle** $\overline{\mathsf{Enc_1}}(k, m)$
	$c_2 \leftarrow \mathtt{Enc}(k'_0, \mathtt{Enc}(k_1, y_0))$	$c \leftarrow \mathtt{Enc}(k, \mathtt{Enc}(k'_1, m))$
	$c_3 \leftarrow \mathtt{Enc}(k'_0, \mathtt{Enc}(k'_1, z_b))$	**return** c
	$c \leftarrow (c_1, c_2, c_3)$	
	return c	**oracle** $\mathsf{H}(\sum \alpha_{x,y} \lvert x, y\rangle)$
		return $\sum \alpha_{x,y} \lvert x, y \oplus \mathsf{H}(x)\rangle$

Figure 5.13: Game G used in the proof of Theorem 5.4.5.

Remark. By removing the quantum random oracle from the games pq2Enc and G and restricting $\mathcal{A}$ to be classical, we obtain a classical security proof for the security under double encryption, yielding essentially the detailed security proof based on the proof sketch of [LP09].

5.5 Summary and Outlook

In this chapter, we gave positive results regarding the post-quantum security of several cryptographic primitives.

Our lifting theorem is an important tool towards the IND-CCA security against quantum adversaries. The typical method to obtain IND-CCA security for public key encryption schemes is via the FO transformation [FO99, FO13], which was shown to also work in the post-quantum setting [HHK17]. Application of the post-quantum FO transformation requires the underlying PKE scheme to be pqIND-CPA secure, which is exactly what our lifting theorem provides. Different variants of the FO transformation [SXY18, JZC+18, HKSU20] achieve tighter post-quantum bounds by using an additional property called disjoint simulatability. An open question is how our lifting theorem relates to this property.

To the best of our knowledge, Sʟae is the first sponge-based AEAD scheme that comes with an explicit proof for its post-quantum security. Since the proof essentially boils down to the post-quantum security of the sponge-based primitives SʟFunc, SPrg, and SvHash, this provides ready-to-use schemes for other scenarios. However, there are some

open questions. First, the proof uses the basic O2H lemma; better bounds can be achieved using its newer variants [AHU19, BHH+19, KSS+20] but it has to be checked whether the additional requirements are met. Second, SLAE and its underlying components are based on the basic sponge construction; the question is whether it easily translates to other constructions such as the duplex construction.

We showed that post-quantum secure deterministic wallets can be constructed from signature schemes with rerandomizable keys. A candidate construction for such a signature scheme from lattices is given in [ADE+20]. Since the results are based on honestly rerandomizable keys, a natural question is what happens when considering arbitrarily rerandomizable keys. Another direction is the extension to threshold primitives [DF90], where signatures are generated from several key shares instead of a single key. The question is whether a generic construction also exists in this case. The challenge is that the individual key shares have to be updated such that they correspond to shares of a randomized key.

We showed that the proof for Yao's GC easily translates to the post-quantum setting. As the main result here, we showed that post-quantum security under double encryption is implied by pqIND-CPA security even in the QROM. This allows, for instance, to use the encryption scheme SLENC which we proved pqIND-CPA in this chapter as well. These results are based on the basic version of Yao's GC. However, there are several techniques that allow for more efficient solutions such as point-and-permute [NPS99], Free-XOR [KS08], garbled row reduction [KMR14, PSSW09, ZRE15]. The exact requirements for these techniques are analysed classically in [GLNP18]; lifting these results to the post-quantum setting allows for more efficient solutions which achieve post-quantum security.

In the upcoming years, a transition to post-quantum secure cryptography is expected, in particular, once the standards from the NIST standardization process [NIST17] are available. On the one hand, this transition includes basic primitives such as public key encryption (also in the scope of [NIST17]) and authenticated encryption schemes. On the other hand, it also includes more advanced primitives and cryptographic protocols. To be confident in these primitives it is crucial to analyse their post-quantum security as we have done in this chapter.

CHAPTER 6

QUANTUM SECURITY

In this chapter, we develop a quantum security notion for public key encryption schemes which allows for a quantum indistinguishability phase. We further provide positive and negative results for public key encryption schemes with respect to this notion and initiate the study of classifying public key encryption schemes in view of applicability of the quantum security notion. The results are based on [GKS21].

Contents

6.1 Quantum Indistinguishability for PKE Schemes 194
6.2 Security Analysis for Real-World PKE Schemes 203
6.3 Classifying Other Public Key Encryption Schemes 212
6.4 Summary and Outlook 216

The scope of this chapter is on the quantum security of cryptographic primitives. This setting assumes that both the adversary and the challenger have quantum computing power and is sometimes referred to as superposition-attack security [DFNS14], QS2 [Gag17], or Q2 [KLLN16b]. Quantum security grants the adversary superposition access to challenger-provided oracles, hereby opening new attack vectors. This is in sharp contrast to the setting of post-quantum cryptography where superposition access is only given to a random oracle while other oracles are restricted to be accessed classically. Note that, for PKE schemes, the adversary can locally encrypt a superposition of messages as it knows the public key. This is also the reason why the adversary gets no explicit encryption oracle for the CPA phase: such an oracle does not provide any additional information.[31]

[31] An exception are resetting attacks, considered in Chapter 4. However, this oracle only reflects the ability to obtain ciphertexts using the particular randomness; the adversary can locally encrypt using any randomness of its choice.

Defining security against quantum adversaries with superposition access to challenger-provided oracles requires some motivation. In the case of random oracles, superposition access has become standard [BDF+11]—even in the post-quantum setting—as they model local computing power of the adversary. In other cases, for instance those considered in [KM12, AR17, KLLN16a], superposition access to keyed cryptographic primitives might look like an artificial extension of the theory.

However, quantum security extends quantum properties to types of attack scenarios not covered in post-quantum security, and at the same time bridges certain security notions from the classical realm to schemes which are meant to run natively on a quantum computer. Some of the reasons why quantum security notions are important to consider are explained in detail in [Gag17]. They basically boil down to five points.

1. To ensure that quantum-resistant classical schemes retain their security even if executed on a quantum computer, possibly in complex environments or protocols where composition should be taken into account.

2. To fix security proofs, where the sole post-quantum security of certain underlying building blocks is not enough to ensure that the whole proof goes through. An example is the need of quantum-secure pseudorandom functions (qPRF) in order to simulate a quantum random oracle [Zha12a], which is a post-quantum concept.

3. To ensure the security of quantum protocols (i.e., meant to run natively on a quantum computer and protect quantum data) when using classical algorithms as building blocks. For example, [Gag17] shows how it is possible to build a secure symmetric quantum encryption scheme (falling into the so-called QS3 domain) by using a quantum-secure symmetric key encryption scheme, but not necessarily a scheme that is post-quantum secure.

4. To consider cases of code obfuscation; for example creating a quantum-resistant PKE scheme by hardcoding a symmetric key into an obfuscated encryption program, a technique known as white-boxing [CEJv03], which is then distributed as a public key.

5. To cover cases of exotic quantum attacks. These include, for instance, quantum fault injection attacks, where a classical device is subject to controlled and artificial physical conditions that induce full or partial quantum behaviour of its hardware (tricking a classical device into being quantum, like in the "frozen smart-card attack" presented in [GHS16]); or cases where a quantum computer is used to run a classical algorithm, but an adversary manages to intercept the intermediate result of the computation before the final measurement meant to produce a classical outcome.

Boneh and Zhandry [BZ13b] initiated the study of quantum security for cryptographic primitives. For signature schemes, they give a security definition that allows the adversary to query the signing oracle on a superposition of messages. For public and symmetric key encryption (PKE and SKE) schemes, on the other hand, they prove that simply allowing the adversary to query a superposition of messages as challenge in a natural way, i.e., giving access to the type-1 unitary similar to the QROM, leads to an unachievable

security notion, called fully-quantum indistinguishability under chosen plaintext attack (fqIND-CPA). Unachievability stems from entanglement between the plaintext register and the ciphertext register. Boneh and Zhandry show how to exploit this entanglement to break the fqIND-CPA security notion irrespectively of the used encryption scheme. To resolve the problem, they propose another security notion (IND-qCPA), which allows the adversary superposition queries in the CPA phase while the challenge messages in the IND phase are restricted to be classical. This notion coincides with the traditional post-quantum security notion for PKE as the adversary can simulate the encryption in superposition, using its local computing power and the public key. For SKE, on the other hand, this yields a notion of quantum security—although the restriction to a classical challenge in this case is clearly a shortcoming.

Gagliardoni et al. [GHS16] overcame this shortcoming for symmetric key encryption schemes by showing how to model a quantum challenge query, while keeping the resulting security notion (qIND-qCPA) still achievable, yet stronger than IND-qCPA. At the heart of their idea lies the use of type-2 operators[32] rather than type-1 operators when encrypting the challenge messages of the adversary. Type-1 operators are the canonical way of implementing a classical function $\mathcal{F}$ on a quantum superposition of input, by mapping the state $|x, y\rangle$ to $|x, y \oplus \mathcal{F}(x)\rangle$, thereby ensuring reversibility for any function $\mathcal{F}$ which is mandatory for any non-measurement quantum operation [NC16]. An important property of type-1 operators is that they create entanglement between the input and output registers, which Boneh and Zhandry exploit to show that fqIND-CPA security is unachievable. In contrast to these operators, type-2 operators work directly on the input register, i.e., they map the state $|x\rangle$ to $|\mathcal{F}(x)\rangle$. Only reversible functions, for instance permutations, can be implemented as type-2 operators, while it is impossible to compute, say, an arbitrary one-way function through a type-2 operator. Gagliardoni et al. [GHS16] observe that SKE schemes act as permutations between the plaintext space and the ciphertext space, which allows to implement the encryption algorithm as a type-2 operator. This observation, in turn, allows to build a solid framework for quantum security in the case of SKE.

Besides these, there are several works on quantum security notions for cryptographic primitives [BZ13b, BZ13a, GHS16, GYZ17, AMRS20, MS16, CEV20, DDKA21]. Yet there is a clear gap when it comes to quantum security notion for public key encryption schemes, as no notion with a quantum indistinguishability phase exists. Here, we close this gap and simultaneously raise several questions for further research in the area of quantum security notions which stem from our results.

Contribution

In this chapter, we present a novel quantum security notion for PKE, provide both achievability results and separation results to the post-quantum security notion for many real-world schemes, and give a general classification of PKE with respect to our security notion.

Our core focus is to extend the results from [GHS16] to the public key scenario. We first formalise the theory of type-2 encryption operators for PKE. For perfectly correct schemes (i.e., schemes which do not suffer from the possibility of decryption failures) we

[32] Also called minimal oracles in [KKVB02].

define the type-2 operator to preserve a randomness register in input and output. Even if such an approach might look strange at first glance, we show that this is the most natural way of defining type-2 operators for PKE schemes. As a next step, we identify a class of PKE schemes—which we call recoverable—where decryption failures can always be avoided given knowledge of the randomness used during encryption, regardless of the actual failure probability of the decryption algorithm. We observe that most real-world partially correct PKE schemes, including many quantum-resistant NIST candidates, are actually of this type. Then, for schemes that are perfectly correct or recoverable, we show how to efficiently construct the type-2 encryption operator. Moreover, we show that for recoverable schemes, this can be done by knowledge of the public key only. This implies, surprisingly, that the adversary can efficiently implement this type-2 operator already in the post-quantum model. This observation marks a substantial difference from the symmetric key case, where the need for type-2 operators is dictated by necessity in order to cover exotic attack models.

Using the theory of type-2 operators developed so far, we give a novel quantum security notion for PKE, that we call quantum indistinguishability under quantum chosen plaintext attack (qIND-qCPA). For a new security notion to be meaningful, two properties are required: first, it has to be achievable, and, second, it has to differ from existing security notions.

We analyse several real-world PKE schemes in respect to our new qIND-qCPA security notion. We show that the canonical LWE-based PKE scheme [Reg05, Reg09] can be attacked for certain parameters. Moving on to code-based schemes, we observe that some schemes encrypt the message using a one-time pad operation which enables an attack. As a concrete example we show that the code-based scheme ROLLO-II [ABD+19] is not qIND-qCPA secure.

However, in practice, most real-world PKE schemes (including the NIST submissions) are used as Key Encapsulation Mechanisms (KEMs) in combination with an SKE scheme, yielding a hybrid PKE-SKE construction. Looking at this canonical hybrid construction then, we show that its qIND-qCPA security mainly depends on the underlying SKE scheme, while the PKE scheme only needs to be secure in the post-quantum sense. Hence, even the code-based PKE scheme ROLLO-II, which as a stand-alone PKE scheme is not qIND-qCPA-secure, can be used to achieve qIND-qCPA security if combined with a qIND-qCPA-secure SKE scheme via the hybrid construction, which is the default way of using it in practice.

We additionally discuss the difficulty of defining type-2 operators—and our related quantum security notion (qIND-qCPA)—for arbitrary schemes that are neither perfectly correct nor recoverable. For this, we study the problem of their general classification and we identify a class of schemes, that we call isometric, that allow to overcome such difficulty. Furthermore, we provide constructions and separation results.

Our results are based on the usage of type-2 operators. This kind of quantum operations is poorly studied in the quantum computing realm, and might therefore look artificial for cryptographic use. We make an effort to expand in a detailed way the formalisation of such operators which, we stress, are only given for functions that are inherently invertible. It is assumed (see for example [GHS16]) that implementing these operators for encryption schemes usually requires knowledge of the secret key. We do not consider this to be a

limitation because in the quantum setting, an honest challenger equipped with the secret key could be allowed to generate particular ciphertext-encoding states which would be hard to compute for an external party: it is therefore necessary to cover this distinction in the preparation of ciphertext states, and type-2 operators do just that. Moreover, as we show in the present work, for many natural PKE schemes, type-2 encryption operators can actually be efficiently implemented by knowledge of the public key only.

Related Work

The study of quantum security under adversarial queries in superposition can be traced back to works such as [DFNS14, Wat09, BDF+11], which explore different settings where this additional adversarial power has an impact on security. However, for the case of signatures and encryption schemes, the first framework going beyond the traditional post-quantum paradigm was given in [BZ13b]. This paradigm was further extended in [GHS16] for symmetric key encryption schemes, and in [AMRS20] for MACs/signatures.

Regarding examples of exotic quantum attacks previously mentioned, it is currently not known whether any of these are feasible at all, but as noted in [Gag17]: (1) if they are feasible, in some cases they do not even require a fully-fledged quantum computer (for example, in the attack from [GHS16] it would be only necessary to produce and detect a Hadamard superposition of messages); and (2) it is already known in the literature that these attacks can be devastating. For example, related-key attacks [RS15], and superposition attacks against Even-Mansour [KM12, BHN+19], FX construction [BNS19, LM17], Feistel networks [IHM+19, KM10, HS18], block ciphers [ATTU16, AR17], and HMAC constructions [KLLN16a].

Qualitatively different, but technically very connected to our setting is the fully quantum setting. This security domain encompasses security notions and constructions for schemes which are natively run on quantum hardware. In the case of fully quantum encryption, these are schemes which are meant to protect quantum, rather than classical data. It turns out that many of the challenges in this area are shared with our scenario of quantum security. In the computational security setting, the first security notions have been provided in [BJ15] for the CPA case, and in [ABF+16] for the CCA1 and semantic security case. These results have been further extended to the CCA2 setting in [AGM18] for the symmetric case, and in [AGM21] for the public key case.

In concurrent and independent work, Chevalier et al. [CEV20] propose alternative quantum security notions for encryption schemes, following the approach by Mossayebi and Schack [MS16] for symmetric key encryption schemes. Their notion and the one presented here are incomparable, as also argued by Carstens et al. [CETU21].

Roadmap

In Section 6.1 we develop the qIND-qCPA security notion which allows for a quantum indistinguishability phase. Furthermore, we give positive and negative results regarding the qIND-qCPA security for real-world public key encryption schemes in Section 6.2. We continue with a general classification of public key encryption schemes regarding the applicability of the qIND-qCPA notion in Section 6.3 and conclude in Section 6.4 with a

summary and an outlook for future research.

6.1 Quantum Indistinguishability for PKE Schemes

In this section, we develop the notion qIND-qCPA security for PKE schemes. It extends the approach from [GHS16] for symmetric key encryption schemes to the public key case, which gets more complex due to the following reasons:

1. PKE schemes are randomised to achieve ciphertext indistinguishability and adversaries must not learn the used randomness, even in the one-time case.

2. when derandomising the encryption procedure and considering the randomness as additional input, there might be collisions (different randomnesses leading to the same ciphertext), hence ensuring reversibility of type-2 operators is not straightforward.

3. many existing schemes, such as lattice- or code-based NIST candidates, suffer from a small decryption failure probability.

In particular, as we will see, there are two main consequences: (1) the inverse of type-2 encryption operators is not generally a type-2 decryption operator; and (2), most interestingly, many type-2 encryption operators can be built efficiently by using only knowledge of the public key. The last point is crucial: it shows that in the PKE case, type-2 encryption operators are much more natural than in the SKE case, and for certain schemes they are actually already covered in the usual notion of post-quantum security. We further show that this includes very relevant schemes, such as the canonical LWE-based scheme used as a blueprint for many NIST submissions. In this section we will do the following:

1. First, we revisit and define formally type-1 operators for PKE, and we show the difference between type-1 encryption and decryption (cf. Section 6.1.1).

2. We define type-2 encryption operators for perfectly correct PKE schemes, and we show that they can be efficiently implemented with knowledge of secret and public key (cf. Section 6.1.2).

3. We define what we call *recoverable* PKE schemes, i.e., schemes that admit an efficient procedure to recover the message given randomness, ciphertext and public key, without the secret key. We show that for such schemes the type-2 encryption operator can be built by only using the public key, even if the scheme suffers from decryption failures (cf. Section 6.1.3).

4. We define the qIND-qCPA security notion for any PKE scheme where one can efficiently build the type-2 encryption operator. This includes in particular perfectly correct and recoverable schemes (cf. Section 6.1.4).

5. Finally, we discuss an extension of the results to the case of using randomness in superposition (cf. Section 6.1.5).

6.1.1 Type-1 Operators for PKE

Recall that, for an arbitrary function $\mathcal{F}\colon \mathcal{X} \to \mathcal{Y}$, the corresponding type-1 operator is the "canonical" way of computing f on a superposition of input through the unitary operator $U_{\mathcal{F}}\colon \mathfrak{H}_{\mathcal{X}} \otimes \mathfrak{H}_{\mathcal{Y}} \to \mathfrak{H}_{\mathcal{X}} \otimes \mathfrak{H}_{\mathcal{Y}}$ defined by: $U_{\mathcal{F}}\colon |x, y\rangle \mapsto |x, y \oplus \mathcal{F}(x)\rangle$. Realising $U_{\mathcal{F}}$ is always efficient if $\mathcal{F}$ is efficiently computable [NC16].

Traditionally, when looking at (deterministic) encryption schemes, the type-1 operator for encryption has been defined as

$$U_{\mathtt{Enc}}\colon |m, y\rangle \mapsto |m, y \oplus \mathtt{Enc}(m)\rangle \ .$$

This is the approach used in [BZ13b, GHS16, MS16]. However, in our case of PKE schemes (which are generally randomised), we have to consider that encryption can be performed locally by the quantum adversary, who therefore has full control not only on the randomness used for encryption (i.e., it is necessary to explicitly derandomise the encryption procedure[33]), but also on the public key used (i.e., it is theoretically possible to compute encryption for a superposition of different public keys). Therefore, the most general definition of a type-1 encryption operator would look like

$$U_{\mathtt{Enc}}\colon |pk, r, m, y\rangle \mapsto |pk, r, m, y \oplus \mathtt{Enc}_{pk}(m; r)\rangle \ .$$

We emphasise that this is indeed the most general and correct way to model the local computational power of a quantum adversary, even in the post-quantum setting. However, for ease of exposition (and also because it would go beyond the traditional meaning of ciphertext indistinguishability), in the present work we do not consider superpositions of public keys, as we assume that the classical public key to be attacked is given to the adversary at the beginning of the game. Hence, we drop the register containing the public key and consider it a parameter of the unitary. This leads us to the following definition.

Definition 6.1.1 (Type-1 Encryption for PKE)**.** *Let* $\textsc{Pke} = (\mathtt{KGen}, \mathtt{Enc}, \mathtt{Dec})$ *be a PKE scheme and let* $(pk, sk) \leftarrow\!\!\$\ \mathtt{KGen}()$*. The type-1 encryption operator for* pk *is the unitary defined by*

$$U^{(1)}_{\mathtt{Enc}_{pk}}\colon |r, m, y\rangle \mapsto |r, m, y \oplus \mathtt{Enc}_{pk}(m; r)\rangle \ .$$

Usually the public key is clear from the context, so we will omit that dependency and just write $U^{(1)}_{\mathtt{Enc}}$. As usual, when there is no ambiguity, we identify the corresponding superoperator acting on mixed states rather than pure states with the same symbol $U^{(1)}_{\mathtt{Enc}}\colon \mathfrak{D}(\mathfrak{H}_{\mathcal{R}} \otimes \mathfrak{H}_{\mathcal{M}} \otimes \mathfrak{H}_{\mathcal{C}}) \to \mathfrak{D}(\mathfrak{H}_{\mathcal{R}} \otimes \mathfrak{H}_{\mathcal{M}} \otimes \mathfrak{H}_{\mathcal{C}})$. By letting the randomness be an input, Definition 6.1.1 allows to encrypt using a superposition of randomnesses, which is fine in the case of the adversary generating ciphertexts himself. Note also that the case of different randomnesses for each message in superposition can be realised by using a single classical randomness and a quantum-secure pseudorandom function [Zha12a], as shown by Boneh and Zhandry [BZ13b].

The type-1 decryption operator is defined analogously to Definition 6.1.1, but with an important difference: the decryption algorithm does not take the randomness used for encryption as input.

[33] This is implicitly considered in [BZ13b, GHS16, MS16], but not explicitly formalised.

Definition 6.1.2 (Type-1 Decryption for PKE)**.** *Let* $\textsc{Pke} = (\texttt{KGen}, \texttt{Enc}, \texttt{Dec})$ *be a PKE scheme and let* $(pk, sk) \leftarrow\!\!\$\ \texttt{KGen}()$*. The type-1 decryption operator for* sk *is the unitary defined by*

$$U^{(1)}_{\texttt{Dec}_{sk}} \colon\ |c, z\rangle \mapsto |c, z \oplus \texttt{Dec}_{sk}(c)\rangle\ .$$

As usual we denote it by $U^{(1)}_{\texttt{Dec}}$, leaving the secret key understood, and when there is no ambiguity with the same symbol we denote the superoperator acting on mixed states also by $U^{(1)}_{\texttt{Dec}} \colon \mathfrak{D}(\mathfrak{H}_{\mathcal{C}} \otimes \mathfrak{H}_{\mathcal{M}}) \to \mathfrak{D}(\mathfrak{H}_{\mathcal{C}} \otimes \mathfrak{H}_{\mathcal{M}})$.

Notice the difference in type-1 encryption and decryption acting on different spaces: this is not surprising, as it is already known that the adjoint of a type-1 encryption operator is not, generally, a type-1 decryption operator. Notice also how both operators are efficiently computable, because `Enc` and `Dec` are efficient algorithms. The difference is that realising $U^{(1)}_{\texttt{Dec}}$ requires knowledge of the secret key sk, while for realising $U^{(1)}_{\texttt{Enc}}$ it is sufficient to know the public key pk.

6.1.2 Type-2 Encryption for PKE

When defining type-2 encryption for PKE schemes, we have to remember that defining these operators only makes sense for functions which are reversible. If a PKE scheme is perfectly correct, then encryption is always reversible if seen as a function of the plaintext, but not necessarily as a function of the randomness. That is because it might be the case that for a given message different randomnesses lead to the same ciphertext. In the context of security games, message and randomness have very different roles anyway, as one is generally chosen by the adversary, while the other is generally chosen by the challenger.

Ultimately, we want to define a type of unitary which generalises the case of arbitrary permutations from plaintext to ciphertext spaces (the same approach as considered in [GHS16]). In order to avoid the issue raised by randomness collisions, we will keep the auxiliary randomness register both in input and output of the circuit. This ensures reversibility of the operator, because given a certain ciphertext and a certain randomness, there is only one possible plaintext which was mapped to that ciphertext (otherwise we would have a decryption failure, and for now we are only considering perfectly correct schemes). So, if the sizes of the plaintext space and the ciphertext space coincide, i.e., there is no ciphertext expansion and thus $\dim(\mathfrak{H}_{\mathcal{M}}) = \dim(\mathfrak{H}_{\mathcal{C}})$, then we can define the corresponding type-2 encryption operator as

$$U^{(2)}_{\texttt{Enc}} \colon\ |r, m\rangle \mapsto |r, \texttt{Enc}_{pk}(m; r)\rangle\ ,$$

where, as usual, the public key pk is implicit in the definition of $U^{(2)}_{\texttt{Enc}}$, i.e., it is a parameter of the unitary operator in question.

In the more general case of message expansion, i.e., $\dim(\mathfrak{H}_{\mathcal{M}}) < \dim(\mathfrak{H}_{\mathcal{C}})$, we use the same approach as in [GHS16]: we introduce an auxiliary register in a complementary space $\mathfrak{H}_{\mathcal{C}-\mathcal{M}}$ that ensures reversibility of the operation, and which is initialised to $|0 \cdots 0\rangle$ during

an honest execution to yield a correct encryption.[34] So we consider a family of unitary superoperators of the form

$$U\colon \mathfrak{D}(\mathfrak{H}_{\mathcal{R}} \otimes \mathfrak{H}_{\mathcal{M}} \otimes \mathfrak{H}_{\mathcal{C}-\mathcal{M}}) \to \mathfrak{D}(\mathfrak{H}_{\mathcal{R}} \otimes \mathfrak{H}_{\mathcal{C}})$$
$$U\colon\ |r, m, y\rangle\langle r, m, y| \mapsto \psi\,,$$

and we define a type-2 encryption operator to be any arbitrary, efficiently computable (purified) representative of the above family such that:

$$U^{(2)}_{\mathtt{Enc}}\colon\ |r, m, 0\cdots 0\rangle \mapsto |r, \mathtt{Enc}_{pk}(m; r)\rangle\ .$$

The choice of the particular representative is irrelevant in our exposition as long as it is efficiently computable and it respects the equation above. However, as already discussed in [GHS16], it might be the case that realising this operator requires knowledge of the secret key, not only of the public key. This finally leads to the following.

Definition 6.1.3 (Type-2 Encryption for PKE)**.** *Let* $\textsc{Pke} = (\mathtt{KGen}, \mathtt{Enc}, \mathtt{Dec})$ *be a perfectly correct PKE scheme, and let* $(pk, sk) \leftarrow\!\!\$\ \mathtt{KGen}()$*. A type-2 encryption operator for* Pke *is an efficiently computable unitary in the family defined by*

$$U^{(2)}_{(\mathtt{Enc},pk,sk)}\colon\ |r, m, 0\cdots 0\rangle \mapsto |r, \mathtt{Enc}_{pk}(m; r)\rangle\ .$$

It will be usually denoted by just $U^{(2)}_{\mathtt{Enc}}$ *when there is no ambiguity.*

It is always possible to find and efficiently sample and implement at least one valid representative for $U^{(2)}_{\mathtt{Enc}}$ given the secret and public keys, by using a conversion circuit of type-1 encryption and decryption operators in a similar way as presented in [GHS16]. We call this the canonical type-2 operator.

Theorem 6.1.4 (Efficient Realisation of Type-2 Encryption)**.** *Let* Pke *be a perfectly correct PKE scheme with* $\textsc{Pke} = (\mathtt{KGen}, \mathtt{Enc}, \mathtt{Dec})$*, and let* $(pk, sk) \leftarrow\!\!\$\ \mathtt{KGen}()$*. Then, there exists an efficient procedure which takes pk and sk as input, and outputs a polynomial-size quantum circuit realising* $U^{(2)}_{\mathtt{Enc}}$*.*

Proof. The explicit circuit of the procedure is shown in Figure 6.1. It uses type-1 encryption and decryption operators as underlying components, which are both efficient with knowledge of the respective keys. □

Notice that realising this canonical type-2 operator requires knowledge of the secret key, even if it is just an encryption operator, but that is fine because as previously mentioned type-2 operators usually require this additional knowledge. We have to make a distinction between the encryption unitary as defined above—a quantum gate modelling local computation of encryption by a party with knowledge of the relevant keys—and the encryption

[34] We denote by $\mathfrak{H}_{\mathcal{C}-\mathcal{M}}$ a Hilbert space such that $\mathfrak{H}_{\mathcal{M}} \otimes \mathfrak{H}_{\mathcal{C}-\mathcal{M}}$ is isomorphic to $\mathfrak{H}_{\mathcal{C}}$. Notice that the opposite case, i.e., $\dim(\mathfrak{H}_{\mathcal{M}}) > \dim(\mathfrak{H}_{\mathcal{C}})$, cannot happen because it would lead to collisions on the ciphertexts and thus introduce decryption failures. Also notice that, as in [GHS16], the case of adversarially-controlled ancilla qubits is left as an open problem.

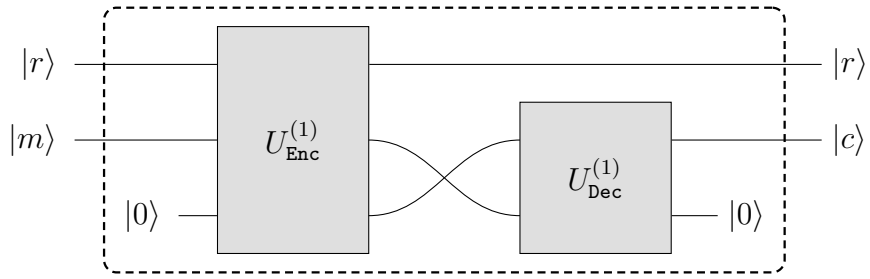

Figure 6.1: Canonical type-2 encryption operator for perfectly correct PKE schemes.

oracle—modelling the interaction of the adversary with such party, usually the challenger. By letting the randomness be an input, Definition 6.1.3 allows to encrypt using a superposition of randomnesses, which is fine in the case of a party generating ciphertexts himself. In our security notion, however, the (honest) challenger will always produce ciphertexts using a (secret) classical randomness not controlled by the adversary $\mathcal{A}$. In the security game, the challenger cannot send the randomness register back to $\mathcal{A}$, because knowledge of the randomness used would trivially break security, even in a classical scenario. But at the same time if the challenger withholds the randomness register, from $\mathcal{A}$'s perspective this would be equivalent to tracing it out, and if the type-2 encryption operator introduces entanglement between ciphertext and randomness output registers, then tracing out the randomness would disturb the ciphertext state.

Luckily, a simple observation solves this dilemma: as we have already discussed, in our oracle case the randomness is chosen by the (honest) challenger during the challenge query, so we can safely model it as classical. Looking at Definition 6.1.3, this means that the output state is always separable as $|r\rangle \otimes |\mathtt{Enc}_{pk}(m;r)\rangle$. Therefore, in our oracle definition the randomness register can be discarded after applying the type-2 encryption without disturbing the ciphertext state. This leads to the following.

Definition 6.1.5 (Type-2 Encryption Oracle)**.** *Let* $\textsc{Pke} = (\mathtt{KGen}, \mathtt{Enc}, \mathtt{Dec})$ *be a PKE scheme and let* $(pk, sk) \leftarrow\!\!\$\ \mathtt{KGen}()$*. The type-2 encryption oracle* Enc *for* pk *is defined by the following procedure:*

oracle $\mathsf{Enc}(|\phi\rangle)$*, where* $|\phi\rangle = \sum_m \alpha_m |m\rangle$

$r \leftarrow\!\!\$\ \mathcal{R}$

$|r\rangle \otimes |\mathtt{Enc}_{pk}(m;r)\rangle \leftarrow U^{(2)}_{\mathtt{Enc}}(|r\rangle\,|\phi\rangle\,|0\cdots 0\rangle)$

trace out $|r\rangle$

return $|\mathtt{Enc}_{pk}(m;r)\rangle$

6.1.3 Recoverable PKE Schemes

Now we introduce a special case of PKE schemes where it is possible to decrypt a ciphertext without knowledge of the secret key, but having access to the randomness used for the encryption instead.[35] These schemes might not be perfectly correct, so the decryption

[35] Concurrently and independently Bellare et al. [BDG20] defined this property in the context of domain separation for random oracles.

procedure may fail on some ciphertext, yet still the recovery procedure will 'decrypt' correctly if the right randomness is provided. We will see in Section 6.2 that many PKE schemes are actually of this type.

Definition 6.1.6 (Recoverable PKE Scheme)**.** *Let* $\textsc{Pke} = (\mathtt{KGen}, \mathtt{Enc}, \mathtt{Dec})$ *be a (not necessarily perfectly correct) PKE scheme. We call* $\textsc{Pke}$ *a* recoverable *PKE scheme if there exists an efficient algorithm* $\mathtt{Rec}\colon \mathcal{PK} \times \mathcal{R} \times \mathcal{C} \to \mathcal{M}$ *such that, for any* $pk \in \mathcal{PK}, r \in \mathcal{R}, m \in \mathcal{M}$*, it holds that*

$$\mathtt{Rec}(pk, r, \mathtt{Enc}_{pk}(m; r)) = m\,.$$

Notice how the recovery procedure will always allow to avoid decryption failures even for schemes which are not perfectly correct. We will sometimes write directly $\textsc{Pke} = (\mathtt{KGen}, \mathtt{Enc}, \mathtt{Dec}, \mathtt{Rec})$ for a recoverable scheme $\textsc{Pke} = (\mathtt{KGen}, \mathtt{Enc}, \mathtt{Dec})$ with recovery algorithm $\mathtt{Rec}$. Given pk, it is of course possible to define a type-1 operator for $\mathtt{Rec}$ in the canonical way.

Definition 6.1.7 (Type-1 Recovery for PKE)**.** *Let* $\textsc{Pke} = (\mathtt{KGen}, \mathtt{Enc}, \mathtt{Dec}, \mathtt{Rec})$ *be a recoverable PKE scheme, and let* $(pk, sk) \leftarrow\!\!\$\, \mathtt{KGen}()$*. The* type-1 recovery operator *for* pk *is the unitary defined by*

$$U^{(1)}_{\mathtt{Rec}_{pk}}\colon\ |r, c, z\rangle \mapsto |r, c, z \oplus \mathtt{Rec}_{pk}(r, c)\rangle\ .$$

As usual we will denote this operator by $U^{(1)}_{\mathtt{Rec}}$ when there is no ambiguity in the choice of pk, and with the same symbol we denote the superoperator acting on mixed states, i.e., $U^{(1)}_{\mathtt{Rec}}\colon \mathfrak{D}(\mathfrak{H}_{\mathcal{R}} \otimes \mathfrak{H}_{\mathcal{C}} \otimes \mathfrak{H}_{\mathcal{M}}) \to (\mathfrak{H}_{\mathcal{R}} \otimes \mathfrak{H}_{\mathcal{C}} \otimes \mathfrak{H}_{\mathcal{M}})$.

Now, the crucial observation is the following: for recoverable PKE schemes, the canonical type-2 encryption operator can be efficiently implemented using only the public key.

Theorem 6.1.8 (Type-2 Encryption Operator for Recoverable Schemes)**.** *Let* $\textsc{Pke}$ *be a recoverable PKE scheme with algorithms* $(\mathtt{KGen}, \mathtt{Enc}, \mathtt{Dec}, \mathtt{Rec})$*, and let* $(pk, sk) \leftarrow\!\!\$\, \mathtt{KGen}()$*. Then, there exists an efficient procedure which only takes* pk *as input, and outputs a polynomial-size quantum circuit realising the canonical operator* $U^{(2)}_{\mathtt{Enc}}$*.*

Proof. The explicit circuit of the procedure is shown in Figure 6.2. It uses type-1 encryption and recovery operators as underlying components, which are both efficient with knowledge of the public key only. Realisation of both these components is independent of the fact whether the scheme has full correctness or not, as the decryption algorithm itself is never used. □

In particular, for recoverable PKE schemes the type-2 encryption operator can be realised locally by a quantum adversary (or a reduction), without need of additional oracle access. This, together with the fact that most real-world PKE schemes are recoverable (as we will see in Section 6.2) shows that type-2 encryption operators are very natural, and unlike in the symmetric key case considered in [GHS16] they also appear implicitly in post-quantum security notions for such schemes.

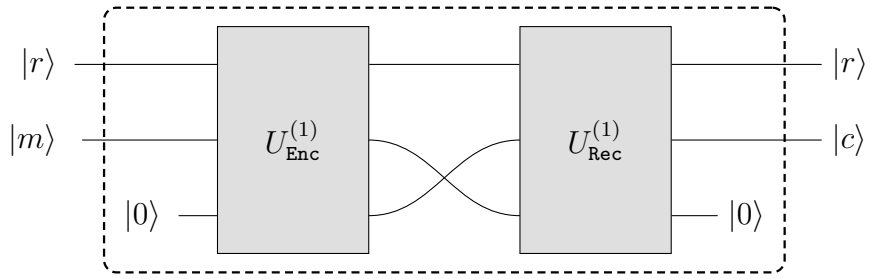

Figure 6.2: Canonical type-2 encryption operator for recoverable PKE schemes.

6.1.4 The qIND-qCPA Security Notion

We are now ready to define the notion of quantum ciphertext indistinguishability under quantum chosen plaintext attack (qIND-qCPA) for PKE schemes which admit an efficient construction of the canonical type-2 encryption operator $U^{(2)}_{\mathtt{Enc}}$. This includes in particular perfectly correct schemes and recoverable schemes.[36] To formally define the game, we have to specify the oracles that the adversary gets access to, using the theory of type-2 operators we have devised so far.

For the challenge (indistinguishability) phase it is pretty straightforward: as in the original qIND security definition for symmetric key encryption, we give the adversary access to a type-2 encryption oracle (cf. Definition 6.1.5) of the PKE scheme in question. Specifically, $\mathcal{A}$ can query oracle $\mathsf{LR\text{-}Enc}$ on two messages $|\phi_0\rangle$ and $|\phi_1\rangle$, potentially in superposition. The oracle will discard (trace out) $|\phi_{1-b}\rangle$, picks a randomness r, applies the type-2 encryption operator to $|r\rangle\,|\phi_b\rangle$, and sends the ciphertext part back to $\mathcal{A}$.

Justifying the use of a type-2 encryption during the challenge phase requires arguments different from the symmetric key case. In the classical IND-CPA game for PKE, the challenger does not even need to know the secret key, as it is not needed for encryption, and we saw already that the secret key is sometimes necessary to implement the canonical type-2 encryption operator. However, in the case of quantum security notions, the challenger can produce ciphertext-encoding quantum states with very different structure depending on whether it knows the secret key or not, thereby leading to different attack models. Type-2 encryption operators in particular are more general in this respect, and allow us to aim for a stronger security notion. Moreover, we also saw how certain schemes, like the recoverable ones, allow to build the type-2 operator using only the public key. Thus, it makes sense for a quantum security notion to include the use of type-2 operators during the challenge phase.

The other question we have to address, which was left unspecified in [GHS16], is about the learning (qCPA) phase. Shall the adversary be able to perform only type-1 encryption operations, or type-2 as well? In the post-quantum setting the answer is simple: it depends on the scheme, e.g., for recoverable schemes both type-1 and type-2 operations are allowed, but in the general case only type-1 operations should. Instead, in the case of quantum security notions that we are considering, the answer is less straightforward.

[36] As we will see, these cover all the interesting cases in practice, although there might be other classes of schemes which allow an efficient construction of $U^{(2)}_{\mathtt{Enc}}$; we address the general case in Section 6.3.

For recoverable schemes again there is no difference, as the adversary can implement both types of operators locally. But for general schemes there might be a difference, and there may exist non-recoverable PKE schemes which become insecure when giving oracle access to a type-2 encryption operator during the learning phase.

In our definition of qIND-qCPA we opt for giving to the adversary as much power as possible, hence explicitly giving access to a type-2 encryption oracle when dealing with non-recoverable schemes, both in the learning and challenge phases. The reason for this choice is twofold. First, this allows us to aim for potentially stronger security notions. Second, remember that, classically, chosen plaintext attacks do not only model the case where the adversary can compute ciphertexts himself (as in the case of PKE), but also scenarios where the adversary can "trick" an honest encryptor in providing certain ciphertexts (as in the case of IND-CPA security for symmetric key encryption). In the quantum PKE setting, there is a difference whether these ciphertexts are computed locally by the adversary or obtained by the challenger through "trickery" (including scenarios already considered in [GHS16], such as quantum side-channel attacks, quantum obfuscation, etc.), because the challenger has knowledge of the secret key, and is therefore capable of generating type-2 ciphertexts even if the scheme is non-recoverable. So, giving the adversary access to the type-2 encryption oracle seems to be the safe choice.

These considerations finally lead to the following.

Definition 6.1.9 (qIND-qCPA Security)**.** *Let* $\textsc{Pke} = (\texttt{KGen}, \texttt{Enc}, \texttt{Dec})$ *be a public key encryption scheme and the game* qINDqCPA *be as defined in Figure 6.3. For any adversary* $\mathcal{A}$*, its corresponding* qIND-qCPA *advantage is given by*

$$\mathbf{Adv}^{\mathsf{qINDqCPA}}_{\textsc{Pke}}(\mathcal{A}) := \left|2\Pr[\mathsf{qINDqCPA}^{\mathcal{A}} \Rightarrow 1] - 1\right| .$$

Game qINDqCPA

$b \leftarrow_{\$} \{0,1\}$
$(pk, sk) \leftarrow_{\$} \texttt{KGen}()$
$b' \leftarrow \mathcal{A}^{\mathsf{LR\text{-}Enc},\mathsf{Enc}}(pk)$
return $(b' = b)$

oracle LR-Enc$(|\phi_0\rangle, |\phi_1\rangle)$, where $|\phi_i\rangle = \sum_m \alpha_{m,i} |m\rangle$

trace out $|\phi_{1-b}\rangle$
$r \leftarrow_{\$} \mathcal{R}$
$|r\rangle |c\rangle \leftarrow U^{(2)}_{\texttt{Enc}} (|r\rangle |\phi_b\rangle |0 \cdots 0\rangle)$
trace out $|r\rangle$
return $|c\rangle$

oracle Enc$(|\phi\rangle)$, where $|\phi\rangle = \sum_m \alpha_m |m\rangle$

$r \leftarrow_{\$} \mathcal{R}$
$|r\rangle |c\rangle \leftarrow U^{(2)}_{\texttt{Enc}} (|r\rangle |\phi\rangle |0 \cdots 0\rangle)$
trace out $|r\rangle$
return $|c\rangle$

Figure 6.3: Security game qINDqCPA.

It is easy to show that the above notion is at least as strong as the post-quantum security notion of IND-qCPA for PKE schemes introduced in [BZ13b].

Theorem 6.1.10 (qIND-qCPA ⇒ IND-qCPA)**.** *Let* $\text{PKE} = (\texttt{KGen}, \texttt{Enc}, \texttt{Dec})$ *be a PKE scheme. For any adversary* $\mathcal{A}$*, it holds that*

$$\mathbf{Adv}^{\mathsf{INDqCPA}}_{\text{PKE}}(\mathcal{A}) \leq \mathbf{Adv}^{\mathsf{qINDqCPA}}_{\text{PKE}}(\mathcal{A})\,.$$

Proof. We show that any adversary $\mathcal{A}$ wins the game qINDqCPA with at least the same probability of winning the game INDqCPA. The differences with game qINDqCPA (cf. Figure 6.3) are:

1. In the game INDqCPA, $\mathcal{A}$ does not get oracle access to Enc. The reason is that the adversary knows the public key which allows to encrypt locally. Hence, when switching to qINDqCPA, the winning probability cannot decrease, because the power of the adversary is augmented by the type-2 oracle.

2. In the game INDqCPA, $\mathcal{A}$ is restricted to classical challenge messages m_0, m_1. When switching to game qINDqCPA, the adversary will simply use the corresponding quantum states $|m_0\rangle, |m_1\rangle$ as challenge messages instead, and will measure the quantum ciphertext received by the oracle.

Notice in fact that, since the randomness r in the qINDqCPA challenge query is classical, the type-2 operator $U^{(2)}_{\texttt{Enc}}$ will produce a ciphertext state which is just a classical ciphertext encoded as a basis state $|c = \texttt{Enc}_{pk}(m; r)\rangle$. In other words, quantum messages are more generic than classical messages or, to put it differently, classical messages are a very special case of quantum messages, and hence again the power of the adversary is not diminished when switching to game qINDqCPA.

More precisely, a reduction $\mathcal{R}$ would simply run adversary $\mathcal{A}$ on the same public key. When $\mathcal{A}$ outputs its two classical messages, $\mathcal{R}$ asks the corresponding quantum states two its oracle LR-Enc and sends the response back to $\mathcal{A}$. In the end, $\mathcal{R}$ simply outputs what $\mathcal{A}$ outputs. □

6.1.5 The Role of Randomness Superposition

In this section we discuss the possibility of having superposition of randomness in the type-2 challenge query. So far, we have only considered the case of classical randomness, as this is chosen by the (honest) challenger. But one could consider scenarios where the adversary can somehow trick the challenger into using a superposition of randomness in the challenge query. Here, we discuss two possible ways to deal with this issue, one of which turns out to be unachievable while the other yields a notion equivalent to the one we propose in Section 6.1.4.

Assume that the challenger chooses a superposition of randomness to encrypt one of the messages chosen by the adversary. Following our security experiment, the challenger would keep the randomness register and merely send the ciphertext register to the adversary. The crucial observation is that the registers containing the randomness and the ciphertext are now entangled. As observed in [GHS16], withholding the randomness register is equivalent to measuring it from the point of view of the adversary. This means that the approach would in fact be equivalent to our security notion using a classical randomness.

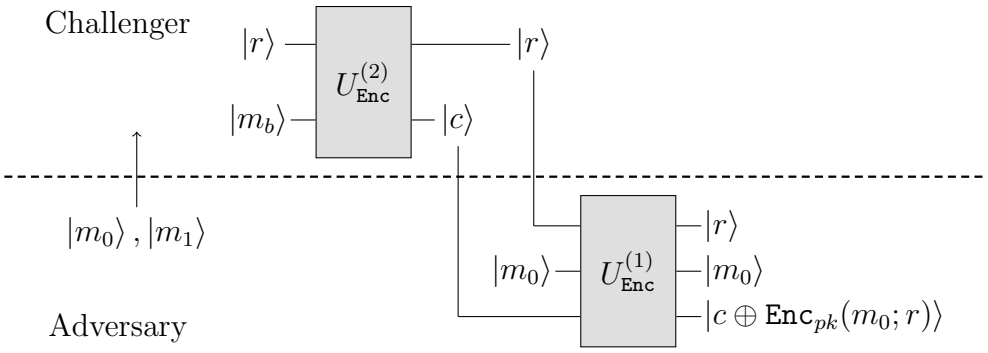

Figure 6.4: Generic attack against superposition of randomness.

Alternatively, to prevent the aforementioned issue of entanglement between the challenger and the adversary, we might let the challenger send the randomness register to the adversary. However, the resulting security notion is unachievable as it would allow the adversary to always distinguish encryptions. We illustrate this with the following attack. First, the adversary chooses two distinct classical messages m_0, m_1, and executes the qIND challenge query with these two. Upon receiving the ciphertext register and the randomness register, the adversary evaluates (locally) the type-1 encryption operator initialising the input register with $|m_0\rangle$, the randomness register with the randomness state received from the challenger, and the ancilla register with the received ciphertext. Finally, the adversary measures the ciphertext register output of the type-1 encryption operator: if it measures 0, then it outputs $b = 0$, otherwise outputs $b = 1$. The circuit is depicted in Figure 6.4. The attack works because, if $b = 0$, then the adversary will compute the same ciphertext as the challenger, hence the output register of the type-1 encryption will be $|0\rangle$; on the other hand, if $b = 1$, a random value will be observed instead. Clearly, this results in output states that the adversary can distinguish with overwhelming probability.

6.2 Security Analysis for Real-World PKE Schemes

We analyse the qIND-qCPA security of several real-world public key encryption schemes. We start with the canonical LWE-based PKE scheme in Section 6.2.1, followed by the code-based PKE scheme ROLLO-II in Section 6.2.2. The hybrid encryption scheme is analysed in Section 6.2.3 while Section 6.2.4 concludes with a discussion of these results.

6.2.1 Lattice-Based PKE

In this section we analyse the canonical LWE-based public key encryption scheme due to Regev [Reg05, Reg09] with respect to our qIND-qCPA security notion.

Recall that qIND-qCPA security can only be defined for schemes which admit an efficient realisation of a type-2 encryption operator. Showing this realisation for the canonical LWE scheme is hence our first goal.

Lemma 6.2.1. *The canonical LWE-based PKE scheme* $\text{PKE} = (\texttt{KGen}, \texttt{Enc}, \texttt{Dec})$*, shown in Figure 2.28, is recoverable as from Definition 6.1.6.*

Proof. To prove the statement, we have to specify the algorithm `Rec`, that is introduced in Definition 6.1.6. Its input is a public key $pk = (\mathsf{a}, \mathsf{b})$, a randomness r, and a ciphertext $c = (c_1, c_2)$ such that c corresponds to the encryption of a message m, using the public key pk and randomness r. The algorithm `Rec` proceeds as follows. Given the randomness r, it obtains the same values e_1, e_2, and d that have been derived from r during encryption and outputs

$$\begin{aligned}\mathsf{Decode}(c_1 - \mathsf{bd} - \mathsf{e}_1) &= \mathsf{Decode}(\mathsf{bd} + \mathsf{e}_1 + \mathsf{Encode}(m) - \mathsf{bd} - \mathsf{e}_1) \\ &= \mathsf{Decode}(\mathsf{Encode}(m)) = m\,.\end{aligned}$$

This concludes the proof. □

Here we give an attack against the canonical LWE-based scheme for the case $q = 2$.

Theorem 6.2.2 (Attack Against Canonical LWE Scheme for $q = 2$)**.** *Let* PKE *be the canonical LWE-based PKE scheme shown in Figure 2.28 defined over* $\mathbb{Z}_q$ *with* $q = 2$*. Then, there exists an efficient adversary* $\mathcal{A}$ *such that*

$$\mathbf{Adv}^{\mathsf{qINDqCPA}}_{\text{PKE}}(\mathcal{A}) = 1\,.$$

Proof. First of all recall that, because $q = 2$, group elements are just represented as bits. This means that $\tau = 0$ (there is neither a padding nor a message expansion) and $\mathsf{Bit}(\mathsf{Encode}(m)) = \pi(m)$. Moreover, addition is performed by XORing elements bitwise. The adversary $\mathcal{A}$ prepares two states

$$|\phi_0\rangle = H\,|0\cdots 0\rangle = \sum_m \frac{1}{\sqrt{2^\mu}}\,|m\rangle \quad \text{and}$$

$$|\phi_1\rangle = H\,|1\cdots 1\rangle = \sum_m \frac{(-1)^{\mathsf{parity}(m)}}{\sqrt{2^\mu}}\,|m\rangle\ .$$

Then $\mathcal{A}$ sends $|\phi_0\rangle, |\phi_1\rangle$ to its oracle LR-Enc. Upon receiving the challenge ciphertext $|c\rangle = |c_1\rangle\,|c_2\rangle$, $\mathcal{A}$ performs the following steps: (1) trace out $|c_2\rangle$, (2) apply $U^{(2)}_{\pi^{-1}}$ to $|c_1\rangle$, where π is the permutation from Lemma 2.6.8, (3) perform a measurement in the Hadamard basis, and (4) if the outcome is $0\cdots 0$ output 0, otherwise, output 1.

If the secret bit of game qINDqCPA equals 0, oracle LR-Enc will encrypt $|\phi_0\rangle$, hence the adversary receives the state

$$|c\rangle = \sum_m \frac{1}{\sqrt{2^\mu}}\,|\texttt{Enc}_{pk}(m;r)\rangle = \sum_m \frac{1}{\sqrt{2^\mu}}\,|c_1^{(m)}\rangle \otimes |c_2\rangle\ ,$$

where $c_1^{(m)}$ is the c_1 part of the ciphertext (cf. Figure 2.28) related to superposition element m. Note that the second part c_2 is independent of the underlying message and hence the two registers are unentangled. After tracing out the second, $\mathcal{A}$ gets the state

$$\sum_m \frac{1}{\sqrt{2^\mu}}\,|c_1^{(m)}\rangle = \sum_m \frac{1}{\sqrt{2^\mu}}\,|\mathsf{u} + \mathsf{Encode}(m)\rangle = \sum_m \frac{1}{\sqrt{2^\mu}}\,|\mathsf{u} + \pi(m)\rangle\ ,$$

for some unknown u representing the LWE sample that is added to the encoded message (cf. Figure 2.28). Using the bit representation of u and writing $\pi^{-1}(u) = w$, from Lemma 2.6.8 the state above can be written as

$$\sum_m \frac{1}{\sqrt{2^{|\mu}}} |u \oplus \pi(m)\rangle = \sum_m \frac{1}{\sqrt{2^\mu}} |\pi(w \oplus m)\rangle \ .$$

Applying the inverse permutation $U^{(2)}_{\pi^{-1}}$ to the state then yields the state

$$\sum_m \frac{1}{\sqrt{2^\mu}} |w \oplus m\rangle = H |0 \cdots 0\rangle \ .$$

Measurement of the state in the Hadamard basis yields $0 \cdots 0$ with probability 1, i.e., it follows that

$$\Pr[\mathcal{A}^{\mathsf{qINDqCPA}} \Rightarrow 0 \,|\, b = 0] = 1 \,.$$

If the secret bit of game $\mathsf{qINDqCPA}$ equals 1, oracle $\mathsf{LR\text{-}Enc}$ will encrypt $|\phi_1\rangle$ and $\mathcal{A}$ receives the state

$$|c\rangle = \sum_m \frac{1}{\sqrt{2^\mu}} |\mathtt{Enc}_{pk}(m; r)\rangle = \sum_m \frac{(-1)^{\mathsf{parity}(m)}}{\sqrt{2^\mu}} |c_1^{(m)}\rangle \otimes |c_2\rangle \ ,$$

where, again, $c_1^{(m)}$ is the c_1 part of the ciphertext related to superposition element m and c_2 is an unentangled state. This yields, after tracing out $|c_2\rangle$, the following state

$$\begin{aligned} \sum_m \frac{(-1)^{\mathsf{parity}(m)}}{\sqrt{2^\mu}} |c_1^{(m)}\rangle &= \sum_m \frac{(-1)^{\mathsf{parity}(m)}}{\sqrt{2^\mu}} |\mathsf{u} + \mathsf{Encode}(m)\rangle \\ &= \sum_m \frac{(-1)^{\mathsf{parity}(m)}}{\sqrt{2^\mu}} |\mathsf{u} + \pi(m)\rangle \ , \end{aligned}$$

where u denotes the LWE sample that is added as described in Figure 2.28. The above state, using bit representation of u and denoting $\pi^{-1}(u)$ by w, equals the following state

$$\sum_m \frac{(-1)^{\mathsf{parity}(m)}}{\sqrt{2^{|\mu}}} |u \oplus \pi(m)\rangle = \sum_m \frac{(-1)^{\mathsf{parity}(m)}}{\sqrt{2^\mu}} |\pi(w \oplus m)\rangle \ .$$

Application of $U^{(2)}_{\pi^{-1}}$ then yields

$$\sum_m \frac{(-1)^{\mathsf{parity}(m)}}{\sqrt{2^\mu}} |w \oplus m\rangle = \sum_m \frac{(-1)^{\mathsf{parity}(w \oplus m)}}{\sqrt{2^\mu}} |m\rangle \ ,$$

i.e., a superposition over all messages with equal amplitudes, half of which being positive and the other half being negative. Hence, a measurement in the Hadamard basis yields $0 \cdots 0$ with probability 0, thus

$$\Pr[\mathcal{A}^{\mathsf{qINDqCPA}} \Rightarrow 0 \,|\, b = 1] = 0 \,.$$

Combining the above finally yields

$$\mathbf{Adv}^{\mathsf{qINDqCPA}}_{\mathrm{PKE}}(\mathcal{A}) = \Big|\Pr[\mathcal{A}^{\mathsf{qINDqCPA}} \Rightarrow 0 \,|\, b = 0] - \Pr[\mathcal{A}^{\mathsf{qINDqCPA}} \Rightarrow 0 \,|\, b = 1]\Big| = 1 - 0 = 1 \,.$$

This concludes the proof. □

6.2.2 Code-based PKE

In this section we analyse the code-based PKE scheme ROLLO-II [ABD+19] with respect to our qIND-qCPA security notion. It turns out that, due to the one-time pad encryption, ROLLO-II is not qIND-qCPA-secure.

We first show that ROLLO-II is recoverable, and hence admits a qIND-qCPA security definition.

Lemma 6.2.3. *The code-based PKE scheme ROLLO-II, shown in Figure 2.30, is recoverable as from Definition 6.1.6.*

Proof. To prove the statement, we have to specify the algorithm $\texttt{Rec}$ that is introduced in Definition 6.1.6. Its input is a public key $pk = \mathbf{h}$, a randomness r, and a ciphertext $c = (c_1, c_2)$, such that c corresponds to the encryption of a message m, using the public key pk and randomness r. The algorithm $\texttt{Rec}$ proceeds as follows. Given the randomness r, it obtains the same values $\mathbf{e}_1$ and $\mathbf{e}_2$ that have been derived from r during encryption. It then computes $\mathsf{H}(\mathsf{Supp}(\mathbf{e}_1, \mathbf{e}_2))$ and outputs $c_1 \oplus \mathsf{H}(\mathsf{Supp}(\mathbf{e}_1, \mathbf{e}_2))$. □

At this point, we would like to point out that the code-based PKE schemes which underlie the NIST proposals BigQuake [CBB+17], HQC [AAB+19a], and RQC [AAB+19b] are recoverable as well.

We give an explicit attack against the qIND-qCPA security of ROLLO-II. It is a Hadamard distinguisher that exploits the fact that the message is essentially encrypted using a one-time pad (ciphertext part c_1 in Figure 2.30).

Theorem 6.2.4. *Let* PKE *be the code-based public key encryption scheme ROLLO-II shown in Figure 2.30. Then, there exists an efficient adversary $\mathcal{A}$ such that*

$$\mathbf{Adv}^{\mathsf{qINDqCPA}}_{\mathrm{PKE}}(\mathcal{A}) = 1\,.$$

Proof. The adversary $\mathcal{A}$ prepares two states

$$|\phi_0\rangle = H\,|0\cdots 0\rangle = \sum_m \frac{1}{\sqrt{2^\mu}}\,|m\rangle \qquad \text{and}$$
$$|\phi_1\rangle = H\,|1\cdots 1\rangle = \sum_m \frac{(-1)^{\mathsf{parity}(m)}}{\sqrt{2^\mu}}\,|m\rangle\,.$$

Then, $\mathcal{A}$ queries $|\phi_0\rangle, |\phi_1\rangle$ to its oracle $\mathsf{LR\text{-}Enc}$. Upon receiving the challenge ciphertext $|c\rangle = |c_1\rangle\,|c_2\rangle$, $\mathcal{A}$ traces out $|c_2\rangle$ and measures the resulting state in the Hadamard basis. If the measurement outcome is $0\cdots 0$, $\mathcal{A}$ outputs 0, otherwise, $\mathcal{A}$ outputs 1.

If the secret bit of game $\mathsf{qINDqCPA}$ equals 0, oracle $\mathsf{LR\text{-}Enc}$ will encrypt $|\phi_0\rangle$, hence the adversary receives the state

$$|c\rangle = \sum_m \frac{1}{\sqrt{2^\mu}}\,|\texttt{Enc}_{pk}(m;r)\rangle = \sum_m \frac{1}{\sqrt{2^\mu}}\,|c_1^{(m)}\rangle \otimes |c_2\rangle\,,$$

where $c_1^{(m)}$ is the c_1 part of the ciphertext (cf. Figure 2.30) related to superposition element m, while the second part c_2 is independent of the underlying message. Hence, the two corresponding registers are unentangled, and after tracing out the second, $\mathcal{A}$ gets the state

$$\sum_m \frac{1}{\sqrt{2^\mu}} |c_1^{(m)}\rangle = \sum_m \frac{1}{\sqrt{2^\mu}} |m \oplus \mathsf{H}(\mathsf{E})\rangle = \sum_m \frac{1}{\sqrt{2^\mu}} |m\rangle \ .$$

Measurement in the Hadamard basis yields $0 \cdots 0$ with probability 1, i.e., it follows that

$$\Pr[\mathcal{A}^{\mathsf{qINDqCPA}} \Rightarrow 0 \,|\, b = 0] = 1 \,.$$

If the secret bit of game qINDqCPA equals 1, oracle LR-Enc will encrypt $|\phi_1\rangle$ and $\mathcal{A}$ receives the state

$$|c\rangle = \sum_m \frac{(-1)^{\mathsf{parity}(m)}}{\sqrt{2^\mu}} |\mathtt{Enc}_{pk}(m;r)\rangle = \sum_m \frac{(-1)^{\mathsf{parity}(m)}}{\sqrt{2^\mu}} |c_1^{(m)}\rangle \otimes |c_2\rangle \ ,$$

where, again, $c_1^{(m)}$ is the c_1 part of the ciphertext (cf. Figure 2.30) related to superposition element m, while the second part c_2 is independent of the underlying message. Hence, the two corresponding registers are unentangled. Let R denote $\mathsf{H}(\mathsf{E})$, then after tracing out the second, $\mathcal{A}$ gets the state

$$\sum_m \frac{(-1)^{\mathsf{parity}(m)}}{\sqrt{2^\mu}} |c_1^{(m)}\rangle = \sum_m \frac{(-1)^{\mathsf{parity}(m)}}{\sqrt{2^\mu}} |m \oplus R\rangle = \sum_m \frac{(-1)^{\mathsf{parity}(m \oplus R)}}{\sqrt{2^\mu}} |m\rangle \ .$$

This means that $\mathcal{A}$ gets a superposition over all μ bit strings which half of the amplitudes being positive and half being negative but all amplitudes being equal in their absolute values. Hence, measurement in the Hadamard basis yields $0 \cdots 0$ with probability 0. This yields

$$\Pr[\mathcal{A}^{\mathsf{qINDqCPA}} \Rightarrow 0 \,|\, b = 1] = 0 \,.$$

By collecting the bounds we obtain

$$\mathbf{Adv}^{\mathsf{qINDqCPA}}_{\mathrm{PKE}}(\mathcal{A}) = \Big|\Pr[\mathcal{A}^{\mathsf{qINDqCPA}} \Rightarrow 0 \,|\, b = 0] - \Pr[\mathcal{A}^{\mathsf{qINDqCPA}} \Rightarrow 0 \,|\, b = 1]\Big| = 1 - 0 = 1 \,.$$

This concludes the proof. □

We note that the attack also works against the code-based scheme BigQuake [CBB+17] which uses the same one-time pad approach to encrypt the message.

6.2.3 Hybrid Encryption

In this section we analyse the canonical hybrid encryption scheme with respect to our qIND-qCPA security notion. We show that its security mainly depends on the underlying symmetric key encryption scheme.

Below we show that the canonical hybrid encryption scheme is recoverable. Given the randomness, the used one-time key can be obtained, which allows to decrypt the ciphertext part that contains the message. We emphasise that the hybrid encryption scheme is recoverable even if the underlying PKE scheme is not recoverable.

Lemma 6.2.5. *The canonical hybrid encryption scheme* $\text{PKE} = (\texttt{KGen}, \texttt{Enc}, \texttt{Dec})$*, shown in Figure 2.33, is recoverable as from Definition 6.1.6.*

Proof. To prove the statement, we have to specify the algorithm `Rec` that is introduced in Definition 6.1.6. Its input is a public key pk, a randomness r, and a ciphertext $c = (c_1, c_2)$, such that c corresponds to the encryption of a message m, using the public key pk and randomness r. The algorithm `Rec` proceeds as follows. Given the randomness r, it obtains r_1, r_2, and r_3, which have been derived from r during encryption. It then obtains $k \leftarrow r_1$ and outputs $\texttt{Dec}_k^S(c_1)$. This concludes the proof. □

We now turn our attention towards the quantum security of the hybrid encryption scheme. It turns out that the quantum security depends on the underlying SKE scheme, while the underlying PKE scheme merely requires post-quantum security. This is formalised in the theorem below.

Theorem 6.2.6 (Quantum Security of Hybrid Encryption)**.** *Let* $\text{PKE} = (\texttt{KGen}, \texttt{Enc}, \texttt{Dec})$ *be the hybrid encryption scheme built from an SKE scheme* $\Sigma^S = (\texttt{Enc}^S, \texttt{Dec}^S)$ *and a PKE scheme* $\Sigma^P = (\texttt{KGen}^P, \texttt{Enc}^P, \texttt{Dec}^P)$*, as shown in Figure 2.33. For any adversary* $\mathcal{A}$ *against* Σ*, there exist adversaries* $\mathcal{A}_{\mathsf{pke}}$ *and* $\mathcal{A}_{\mathsf{ske}}$ *against* Σ^P *and* Σ^S*, respectively, such that*

$$\mathbf{Adv}^{\mathsf{qIND qCPA}}_{\text{PKE}}(\mathcal{A}) \leq 2\,\mathbf{Adv}^{\mathsf{IND qCPA}}_{\Sigma^P}(\mathcal{A}_{\mathsf{pke}}) + \mathbf{Adv}^{\mathsf{qIND}}_{\Sigma^S}(\mathcal{A}_{\mathsf{ske}})\,.$$

Proof. The proof uses four games $\mathsf{G}_{0,0}$, $\mathsf{G}_{1,0}$, $\mathsf{G}_{1,1}$, and $\mathsf{G}_{0,1}$ described in Figure 6.5. Game $\mathsf{G}_{0,b}$ is the game $\mathsf{qIND qCPA}$ instantiated with PKE and secret bit b. Game $\mathsf{G}_{1,b}$ differs in that a randomly chosen key k' is encrypted using the underlying PKE scheme Σ^P rather than key k which is used to encrypt the message. It holds that

$$\begin{aligned}
\mathbf{Adv}^{\mathsf{qIND qCPA}}_{\text{PKE}}(\mathcal{A}) &= \left|\Pr[\mathcal{A}^{\mathsf{qIND qCPA}} \Rightarrow 0 \,|\, b = 0] - \Pr[\mathcal{A}^{\mathsf{qIND qCPA}} \Rightarrow 0 \,|\, b = 1]\right| \\
&= \left|\Pr[\mathcal{A}^{\mathsf{G}_{0,0}} \Rightarrow 0] - \Pr[\mathcal{A}^{\mathsf{G}_{0,1}} \Rightarrow 0]\right| \\
&\leq \left|\Pr[\mathcal{A}^{\mathsf{G}_{0,0}} \Rightarrow 0] - \Pr[\mathcal{A}^{\mathsf{G}_{1,0}} \Rightarrow 0]\right| \\
&\quad + \left|\Pr[\mathcal{A}^{\mathsf{G}_{1,0}} \Rightarrow 0] - \Pr[\mathcal{A}^{\mathsf{G}_{1,1}} \Rightarrow 0]\right| \\
&\quad + \left|\Pr[\mathcal{A}^{\mathsf{G}_{1,1}} \Rightarrow 0] - \Pr[\mathcal{A}^{\mathsf{G}_{0,1}} \Rightarrow 0]\right| .
\end{aligned}$$

For the first difference, we construct an adversary $\mathcal{A}_{\mathsf{0}}$ against the IND-qCPA security of Σ^P. Upon receiving a public key pk, $\mathcal{A}_{\mathsf{0}}$ runs $\mathcal{A}$ on the same public key. Queries $|\phi\rangle$ by $\mathcal{A}$ are processed locally by $\mathcal{A}_{\mathsf{0}}$. Concretely, $\mathcal{A}_{\mathsf{0}}$ samples $r_1, r_2, r_3 \leftarrow\!\!\$\, \mathcal{R}$ and derives $k \leftarrow r_1$. Then, it applies $U^{(2)}_{\texttt{Enc}_k}$ to $|\phi\rangle$ yielding $|c_1\rangle$. Subsequently, it computes $c_2 \leftarrow \texttt{Enc}_{pk}(k; r_3)$. Note that this step is entirely classical. Finally, $\mathcal{A}_{\mathsf{0}}$ returns $|c\rangle \leftarrow |c_1, c_2\rangle$ to $\mathcal{A}$. For queries $|\phi_0\rangle, |\phi_1\rangle$ to $\mathsf{LR\text{-}Enc}$, $\mathcal{A}_{\mathsf{0}}$ samples $r_1, r_2, r_3 \leftarrow\!\!\$\, \mathcal{R}$, derives $k \leftarrow r_1$, and samples another key $k' \leftarrow\!\!\$\, \mathcal{K}$. It applies $U^{(2)}_{\texttt{Enc}_k}$ to $|\phi_0\rangle$ to obtain $|c_1\rangle$. It queries k, k' to its own (classical) $\mathsf{LR\text{-}Enc}$ to obtain c_2 and sends $|c\rangle \leftarrow |c_1\rangle\,|c_2\rangle$ to $\mathcal{A}$. Eventually, $\mathcal{A}_{\mathsf{0}}$ outputs whatever $\mathcal{A}$ outputs.

It holds that $\mathcal{A}_0$ perfectly simulates $\mathsf{G}_{0,0}$ and $\mathsf{G}_{1,0}$ for $\mathcal{A}$ conditioned on its own challenge bit from game $\mathsf{INDqCPA}$ being 0 and 1, respectively. By outputting the same as $\mathcal{A}$, we get

$$\begin{aligned}\left|\Pr[\mathcal{A}^{\mathsf{G}_{0,0}} \Rightarrow 0] - \Pr[\mathcal{A}^{\mathsf{G}_{1,0}} \Rightarrow 0]\right| &= \left|\Pr[\mathcal{A}_0^{\mathsf{INDqCPA}} \Rightarrow 0 \,|\, b = 0] - \Pr[\mathcal{A}_0^{\mathsf{INDqCPA}} \Rightarrow 0 \,|\, b = 1]\right| \\ &= \mathbf{Adv}_{\Sigma}^{\mathsf{INDqCPA}}(\mathcal{A}_0)\,.\end{aligned}$$

For the second difference we construct adversary $\mathcal{A}_{\mathsf{ske}}$ against the qIND security of Σ^S. At the start, $\mathcal{A}_{\mathsf{ske}}$ computes $(pk, sk) \leftarrow\!\!\$\ \mathtt{KGen}()$ and runs $\mathcal{A}$ on input pk. When $\mathcal{A}$ makes a query $|\phi\rangle$ to Enc, $\mathcal{A}_{\mathsf{ske}}$ samples $r_1, r_2, r_3 \leftarrow\!\!\$\ \mathcal{R}$, derives $k \leftarrow r_1$, and computes $|c_1\rangle$ by applying $U^{(2)}_{\mathtt{Enc}^S}$ to $|\phi\rangle$. Furthermore, $\mathcal{A}_{\mathsf{ske}}$ computes (classically) $c_1 \leftarrow \mathtt{Enc}^P_{pk}(k; r_3)$ and sends $|c\rangle \leftarrow |c_1\rangle\,|c_2\rangle$ back to $\mathcal{A}$. For the challenge query $|\phi_0\rangle\,, |\phi_1\rangle$ by $\mathcal{A}$, $\mathcal{A}_{\mathsf{ske}}$ obtains $|c_1\rangle$ by forwarding the query to its own oracle $\mathsf{LR\text{-}Enc}$ while it locally computes $c_2 \leftarrow \mathtt{Enc}^P_{pk}(k'; r_3)$, for randomly chosen k' and r_3. Then $\mathcal{A}_{\mathsf{ske}}$ sends $|c\rangle \leftarrow |c_1\rangle\,|c_2\rangle$ back to $\mathcal{A}$. When $\mathcal{A}$ terminates by outputting a bit, $\mathcal{A}_{\mathsf{ske}}$ outputs the same bit.

It holds that $\mathcal{A}_{\mathsf{ske}}$ simulates $\mathsf{G}_{1,b}$ for $\mathcal{A}$, where b is the challenge bit from its own game qIND. Hence, we get

$$\begin{aligned}\left|\Pr[\mathcal{A}^{\mathsf{G}_{1,0}} \Rightarrow 0] - \Pr[\mathcal{A}^{\mathsf{G}_{1,1}} \Rightarrow 0]\right| &= \left|\Pr[\mathcal{A}_{\mathsf{ske}}^{\mathsf{qIND}} \Rightarrow 0 \,|\, b = 0] - \Pr[\mathcal{A}_{\mathsf{ske}}^{\mathsf{qIND}} \Rightarrow 0 \,|\, b = 1]\right| \\ &= \mathbf{Adv}_{\Sigma}^{\mathsf{qIND}}(\mathcal{A}_{\mathsf{ske}})\,.\end{aligned}$$

For the third difference, we construct $\mathcal{A}_1$ against the IND-qCPA security of Σ^P, similar to $\mathcal{A}_0$. It runs $\mathcal{A}$ on the same public key pk as it receives. Queries $|\phi\rangle$ are answered locally by choosing $r_1, r_2, r_3 \leftarrow\!\!\$\ \mathcal{R}$, deriving the symmetric key $k \leftarrow r_1$, and computing $|c_1\rangle$ by applying $U^{(2)}_{\mathtt{Enc}^S_k}$ to $|\phi\rangle$. In addition, $\mathcal{A}_1$ computes $c_2 \leftarrow \mathtt{Enc}_{pk}(m; r_3)$ and sends $|c\rangle \leftarrow |c_1\rangle\,|c_2\rangle$ to $\mathcal{A}$. Queries $|\phi_0\rangle\,, |\phi_1\rangle$ are processed as follows. First, $\mathcal{A}_1$ samples $r_1, r_2, r_3 \leftarrow\!\!\$ \mathcal{R}$, derives the symmetric key $k \leftarrow r_1$ as well as another key $k' \leftarrow\!\!\$\ \mathcal{K}$. Then, it computes $|c_1\rangle$ by applying $U^{(2)}_{\mathtt{Enc}^S_k}$ to $|\phi_1\rangle$ and obtains c_2 by querying its own oracle $\mathsf{LR\text{-}Enc}$ on k, k'. Finally, it sends $|c_1\rangle\,|c_2\rangle$ to $\mathcal{A}$. When $\mathcal{A}$ terminates and outputs a bit b', $\mathcal{A}_1$ outputs $1 - b'$.

It holds that $\mathcal{A}_1$ perfectly simulates $\mathsf{G}_{1,1}$ if its own challenge bit equals 1 and $\mathsf{G}_{0,1}$ if the challenge bit equals 0. This yields

$$\begin{aligned}\Pr[\mathcal{A}^{\mathsf{G}_{1,1}} \Rightarrow 0] - \Pr[\mathcal{A}^{\mathsf{G}_{0,1}} \Rightarrow 0] &\leq \Pr[\mathcal{A}_1^{\mathsf{INDqCPA}} \Rightarrow 1 \,|\, b = 1] - \Pr[\mathcal{A}_1^{\mathsf{INDqCPA}} \Rightarrow 1 \,|\, b = 0] \\ &= \mathbf{Adv}_{\Sigma^P}^{\mathsf{INDqCPA}}(\mathcal{A}_1)\,.\end{aligned}$$

Collecting the bounds and defining $\mathcal{A}_{\mathsf{pke}}$ to be the adversary with the higher advantage among $\mathcal{A}_0$ and $\mathcal{A}_1$ yields

$$\mathbf{Adv}_{\mathrm{PKE}}^{\mathsf{qINDqCPA}}(\mathcal{A}) \leq 2\,\mathbf{Adv}_{\Sigma^P}^{\mathsf{INDqCPA}}(\mathcal{A}_{\mathsf{pke}}) + \mathbf{Adv}_{\Sigma}^{\mathsf{qIND}}(\mathcal{A}_{\mathsf{ske}})\,.$$

This concludes the proof. □

Theorem 6.2.6 reveals that to achieve our qIND-qCPA security notion, we can instantiate the hybrid encryption scheme with a PKE scheme that merely achieves post-quantum security. This allows the usage of ROLLO-II, which, used as a stand-alone PKE scheme, is not qIND-qCPA-secure.

Games $\mathsf{G}_{0,b}, \mathsf{G}_{1,b}$

$b \leftarrow_{\$} \{0,1\}$
$(pk, sk) \leftarrow_{\$} \mathtt{KGen}()$
$b' \leftarrow \mathcal{A}^{\mathsf{LR\text{-}Enc},\mathsf{Enc}}(pk)$

oracle $\mathsf{LR\text{-}Enc}(|\phi_0\rangle, |\phi_1\rangle)$, where $|\phi_i\rangle = \sum_m \alpha_{m,i} |m\rangle$ in $\mathsf{G}_{0,b}$

trace out $|\phi_{1-b}\rangle$
$r_1, r_2, r_3 \leftarrow_{\$} \mathcal{R}$
$k \leftarrow r_1$
$|c_1\rangle = \sum_m \alpha_{m,b} |\mathtt{Enc}_k^S(m; r_2)\rangle \leftarrow U^{(2)}_{\mathtt{Enc}_k^S} |r_2, \phi_b\rangle$
$|c_2\rangle = |\mathtt{Enc}_{pk}(k; r_3)\rangle \leftarrow U^{(2)}_{\mathtt{Enc}_{pk}^P} |r_3, k\rangle$ // this step is entirely classical
trace out $|r_1, r_2, r_3\rangle$
return $|c\rangle \leftarrow |c_1\rangle |c_2\rangle$

oracle $\mathsf{LR\text{-}Enc}(|\phi_0\rangle, |\phi_1\rangle)$, where $|\phi_i\rangle = \sum_m \alpha_{m,i} |m\rangle$ in $\mathsf{G}_{1,b}$

trace out $|\phi_{1-b}\rangle$
$r_1, r_2, r_3 \leftarrow_{\$} \mathcal{R}$
$k \leftarrow r_1$
$k' \leftarrow_{\$} \mathcal{K}$
$|c_1\rangle = \sum_m \alpha_{m,b} |\mathtt{Enc}_k^S(m; r_2)\rangle \leftarrow U^{(2)}_{\mathtt{Enc}_k^S} |r_2, \phi_b\rangle$
$|c_2\rangle = |\mathtt{Enc}_{pk}(k; r_3)\rangle \leftarrow U^{(2)}_{\mathtt{Enc}_{pk}^P} |r_3, k'\rangle$ // this step is entirely classical
trace out $|r_1, r_2, r_3\rangle$
return $|c\rangle \leftarrow |c_1\rangle |c_2\rangle$

oracle $\mathsf{Enc}(|\phi\rangle)$, where $|\phi\rangle = \sum_m \alpha_m |m\rangle$

$r_1, r_2, r_3 \leftarrow_{\$} \mathcal{R}$
$k \leftarrow r_1$
$|c_1\rangle = \sum_m \alpha_m |\mathtt{Enc}_k^S(m; r_2)\rangle \leftarrow U^{(2)}_{\mathtt{Enc}_k^S} |r_2, \phi\rangle$
$|c_2\rangle = |\mathtt{Enc}_{pk}(k; r_3)\rangle \leftarrow U^{(2)}_{\mathtt{Enc}_{pk}^P} |r_3, k\rangle$ // this step is entirely classical
trace out $|r_1, r_2, r_3\rangle$
return $|c\rangle \leftarrow |c_1\rangle |c_2\rangle$

Figure 6.5: Games $\mathsf{G}_{0,0}$, $\mathsf{G}_{0,1}$, $\mathsf{G}_{1,0}$, and $\mathsf{G}_{1,1}$ used in the proof of Theorem 6.2.6.

In the following we show that Theorem 6.2.6 is strict. If the underlying SKE is not qIND-secure, then the resulting hybrid scheme is not qIND-qCPA-secure, irrespectively of the underlying PKE scheme. This is shown in the theorem below. Examples for SKE schemes which are not qIND-secure are given in [GHS16].

Theorem 6.2.7. *Let* $\text{PKE} = (\texttt{KGen}, \texttt{Enc}, \texttt{Dec})$ *be the hybrid encryption scheme built from an SKE scheme* $\Sigma^S = (\texttt{Enc}^S, \texttt{Dec}^S)$ *and a PKE scheme* $\Sigma^P = (\texttt{KGen}^P, \texttt{Enc}^P, \texttt{Dec}^P)$*, as shown in Figure 2.33. Let* $\mathcal{A}_{\mathsf{ske}}$ *be an adversary such that*

$$\mathbf{Adv}^{\mathsf{qIND}}_{\Sigma^S}(\mathcal{A}_{\mathsf{ske}}) = \epsilon\,.$$

Then, there exists an adversary $\mathcal{A}_{\mathsf{pke}}$ *against* PKE *such that*

$$\mathbf{Adv}^{\mathsf{qINDqCPA}}_{\text{PKE}}(\mathcal{A}_{\mathsf{pke}}) \geq \epsilon\,.$$

Proof. We construct the adversary $\mathcal{A}_{\mathsf{pke}}$, which uses adversary $\mathcal{A}_{\mathsf{ske}}$ as subroutine, as follows. When $\mathcal{A}_{\mathsf{ske}}$ outputs its challenge messages $|\phi_0\rangle$ and $|\phi_1\rangle$, $\mathcal{A}_{\mathsf{pke}}$ forwards these to its own challenger. Upon receiving the challenge ciphertext $|c\rangle = |c_1\rangle\,|c_2\rangle$, $\mathcal{A}_{\mathsf{pke}}$ sends $|c_1\rangle$ to $\mathcal{A}_{\mathsf{ske}}$. When $\mathcal{A}$ outputs its guess b', $\mathcal{A}_{\mathsf{pke}}$ outputs b' as its own guess.

It holds that $\mathcal{A}_{\mathsf{pke}}$ perfectly simulates game qIND, with the same challenge bit b, for $\mathcal{A}_{\mathsf{ske}}$. Note that only sending $|c_1\rangle$ to $\mathcal{A}$ does not disturb the state as $|c_2\rangle$, due to being entirely classical, is unentangled and can therefore be discarded without disturbing the state. By outputting the same bit as $\mathcal{A}_{\mathsf{ske}}$, we obtain

$$\mathbf{Adv}^{\mathsf{qINDqCPA}}_{\text{PKE}}(\mathcal{A}_{\mathsf{pke}}) \geq \mathbf{Adv}^{\mathsf{qIND}}_{\Sigma^S}(\mathcal{A}_{\mathsf{ske}}) = \epsilon\,,$$

which proves the claim. □

6.2.4 Discussion

In this section, we gave both positive and negative examples regarding the quantum security of real-world public key encryption schemes. We gave a concrete attack against the canonical LWE-based encryption scheme for the case $q = 2$ and an attack against the code-based encryption scheme ROLLO-II showing that these schemes are qIND-qCPA insecure. However, these results considered that the schemes are used as public key encryption schemes to encrypt the actual message. On the other hand, Theorem 6.2.6 reveals that both ROLLO-II and the canonical LWE-based encryption scheme are sufficient to achieve qIND-qCPA security when used as a key encapsulation mechanism (KEM), together with a quantum-secure SKE scheme. Note that the post-quantum security of ROLLO-II follows from Theorem 5.1.7 in Chapter 5.

The standardisation effort by NIST [NIST17] focuses on the latter scenario. Hence our results show that for these standardised schemes, it is sufficient to achieve post-quantum security in order for the resulting KEM to achieve our stronger, more conservative quantum security notion. At the same time, our results also show that extra cautiousness is necessary when these standardised schemes are deployed directly as PKE schemes in protocols that require quantum security.

6.3 Classifying Other Public Key Encryption Schemes

So far we have built a framework for quantum security of PKE schemes which are perfectly correct or recoverable (or both). The imminent question that follows is about schemes that satisfy neither of these properties. In this section, we initiate the classification of PKE schemes in general, extend our results to other classes of PKE schemes where possible, and point out the obstacles in other cases.

6.3.1 Dealing with Decryption Failures: The General Case

First, we discuss why arbitrary non-correct PKE schemes do not allow, in general, to define a type-2 encryption operator and, consequently, we cannot always define game qINDqCPA as from Figure 6.3. However, we also discuss a possible workaround.

First of all, recall that defining a type-2 operator is only possible for functions that are inherently invertible. Then observe that a $(1-\alpha)$-correct PKE scheme (cf. Definition 2.2.9) could have arbitrary, even overwhelming decryption error α. In the most extreme case, the scheme can be almost identical to a constant function (for example, consider an artificial scheme where every public key pk always encrypts to 0, except for one particular randomness $\bar{r}$ where it produces a correctly decryptable ciphertext instead). In the presence of decryption failures, it is therefore impossible to find a general way to define type-2 operators for encryption, and hence, to define a suitable qIND-qCPA security notion.[37]

We call non-isometric such schemes, where it is simply not possible to define a unitary operator that behaves exactly as from Definition 6.1.3 for any keypair, even if we drop the requirement of efficiency.

Definition 6.3.1 (Non-Isometric Schemes)**.** *Let $\Sigma = (\mathtt{KGen}, \mathtt{Enc}, \mathtt{Dec})$ be a PKE scheme. We say that Σ is* non-isometric *if, for any $(pk, sk) \leftarrow\!\!\$\ \mathtt{KGen}()$, there exists at least a randomness r_{pk} such that the function $m \mapsto \mathtt{Enc}_{pk}(m; r_{pk})$ is non-injective. In particular, for any unitary U acting on the appropriate Hilbert spaces, there exists at least a pair (m_{pk}, r_{pk}) such that*

$$\Pr\left[M(U\,|r_{pk}, m_{pk}, 0\cdots 0\rangle) \to (r_{pk}, \mathtt{Enc}_{pk}(m_{pk}; r_{pk}))\right] < 1\,,$$

where M denotes measurement in the canonical computational basis.

A possible workaround for these non-isometric schemes is to 'enforce' the reversibility of the encryption, obtaining a new type of encryption unitary. Consider what happens if we want to use the type-1 encryption operator (cf. Definition 6.1.1) during the challenge query

$$U^{(1)}_{\mathtt{Enc}_{pk}}\colon\ |r, m, y\rangle \mapsto |r, m, y \oplus \mathtt{Enc}_{pk}(m; r)\rangle\ .$$

As already observed, the randomness r can be understood as classical and discarded by the challenger. However, the other two registers are generally going to be entangled, and both would have to be sent to the adversary for a meaningful quantum notion; this

[37] Recoverable schemes are a special case: they might not be always reversible in the message space only, but they are always reversible in the union of message space and randomness space.

would clearly break security because the message would remain in clear. Withholding the message register would be equivalent to the challenger just measuring it in which case the notion would be equivalent to the post-quantum scenario. We could try to fix this issue by (reversibly) masking the message register sent to the adversary, for example by using a permutation π on the message space drawn uniformly at random. Consider the following unitary

$$U^{(\pi)}_{\mathtt{Enc}_{pk}} : |r, m, y\rangle \mapsto |r, \pi(m), y \oplus \mathtt{Enc}_{pk}(m; r)\rangle \ .$$

It allows to define a new type of quantum challenge query, where the challenger still discards the randomness register after encryption, but sends back the other two registers to the adversary. Notice how, from the adversary's point of view, $\pi(m)$ is a completely random element, and therefore the presence of this additional register does not offer any distinguishing advantage. Moreover, in actual security reductions, the uniformly drawn π can be replaced by a quantum-secure pseudorandom permutation [GHS16], or qPRP in short.

We can hence use these type-π operators to define (for any PKE scheme, including the non-isometric ones) a new indistinguishability game and a related security notion with quantum challenge query. Motivating the use of such operators when modelling security is arguably non-trivial. In certain cases, one could see $\pi(m)$ as some sort of side-channel information given to the adversary, but in general it looks like just an artificial way to enforce reversibility on schemes which are not. Even though we do not study this notion here, we want nevertheless to make a few observations on it.

First of all, notice that such a new security notion cannot be stronger than qIND-qCPA, at least when considering perfectly correct or recoverable schemes. As a separating example, consider the distinguishing attack from Theorem 6.2.7: this will not work any more because of the presence of the entangled $\pi(m)$ register, so that the hybrid scheme might be secure according to the new notion but still qIND-qCPA insecure.

Second, notice how the challenge query resulting from the use of type-π operators reminds of the one given in an alternative quantum indistinguishability notion for secret key encryption schemes proposed by Mossayebi and Schack [MS16] - the difference is basically producing $|m, \mathtt{Enc}_{pk}(\pi(m))\rangle$ instead of $|\pi(m), \mathtt{Enc}_{pk}(m)\rangle$ - which is itself not comparable to qIND-qCPA. This approach has further been studied by Chevalier et al. [CEV20].

6.3.2 Refining the Classification

Now we know how to define qIND-qCPA security of PKE schemes which are perfectly correct or recoverable (or both), and at the same time we know that it is not possible for schemes that are non-isometric. But it turns out we can say more. First of all we make a distinction for those schemes which are isometric: this means that it is possible to define a unitary operator that behaves exactly as a type-2 encryption operator, but we distinguish whether finding and building such operator is efficient or not.

Definition 6.3.2 ((Efficiently) Isometric Schemes)**.** *Let* PKE *be a PKE scheme with* $\text{PKE} = (\mathtt{KGen}, \mathtt{Enc}, \mathtt{Dec})$*. We say that* PKE *is isometric if, for any* $(pk, sk) \leftarrow\!\!\$\ \mathtt{KGen}()$

and for any randomness r the function $m \mapsto \mathtt{Enc}_{pk}(m;r)$ is injective. In particular, there exists a unitary U acting on the appropriate Hilbert spaces, such that for any (m,r)

$$\Pr\left[M(U\,|r,m,0\cdots0\rangle) \to (r,\mathtt{Enc}_{pk}(m;r))\right] = 1\,,$$

where M denotes measurement in the canonical computational base. Furthermore, we say that PKE *is efficiently isometric if U can be efficiently realised.*

Notice how, in general, an isometric scheme is not necessarily efficiently isometric. This is because, unlike for type-1 operators, the efficiency of the `Enc` procedure is only enough to guarantee the existence of a unitary U with the above property, but not its efficiency. Then, notice how a type-2 encryption operator (as from Definition 6.1.3) satisfies the above definition of U, both by construction and by efficiency. In other words, efficiently isometric schemes are exactly all and only those schemes which, by definition, admit an efficient construction of the type-2 encryption operator. Clearly, in particular this includes perfectly correct schemes and recoverable schemes.

Corollary 6.3.3. *Let* PKE *be a PKE scheme. If* PKE *is perfectly correct or recoverable, then it is efficiently isometric.*

Proof. Follows immediately from Theorem 6.1.4 and Theorem 6.1.8 for perfectly correct and recoverable PKE schemes, respectively. □

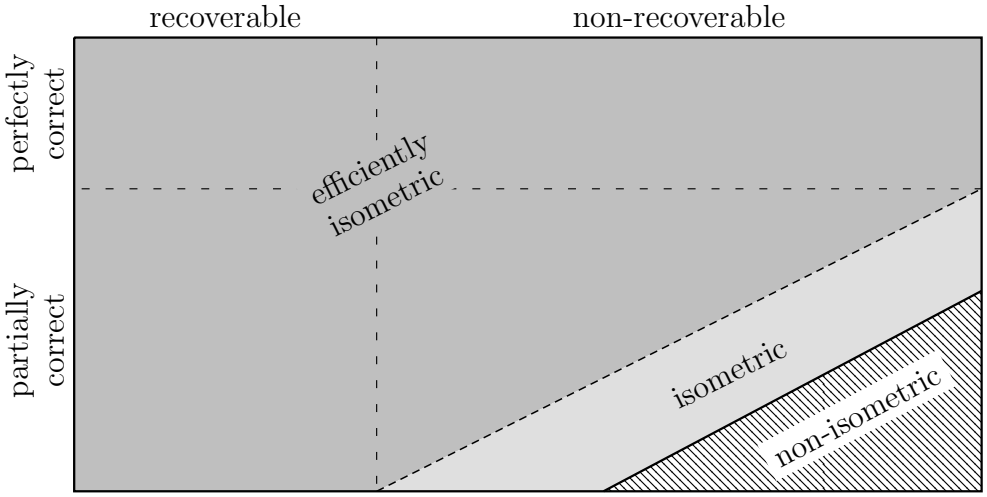

Figure 6.6: Classification of PKE schemes. The qIND-qCPA security notion can be defined for all schemes except those in the shaded area (non-isometric). For efficiently isometric schemes (dark grey area) the type-2 operator can be realised efficiently and we provide concrete circuits for the schemes that are perfectly correct or recoverable. For isometric schemes (light grey area) the type-2 operator can be realised but not efficiently.

The situation is depicted in Figure 6.6. This means that, as from Definition 6.1.9, we can extend the qIND-qCPA security notion not only to recoverable or perfectly correct

schemes, but to all the efficiently isometric ones. For the non-efficient case (arbitrary isometric schemes) the qIND-qCPA notion can still be defined, but its usefulness would be less clear, as it might require unbounded challengers in the security game (and therefore, difficulty in simulating them by efficient reductions when proving the security of a particular scheme). Still, it is useful for impossibility results, i.e., proving that a particular isometric scheme is not qIND-qCPA-secure.

The final question is whether there are examples which fall into the categories we have just defined. We have already mentioned an example of a non-isometric scheme at the beginning of Section 6.3.1, i.e., the almost-constant one. Here, we show a construction of an efficiently isometric scheme that is neither perfectly correct nor recoverable. The construction is given in Figure 6.7: it transforms a recoverable, not perfectly correct encryption scheme (like the canonical LWE-based scheme) by pre-processing the message with a quantum-secure trapdoor permutation [Gag17] before encrypting it, and inverts again the permutation after decryption. The public and secret keys of the trapdoor permutation are embedded in the public and secret key, respectively, of the resulting scheme. It works because the permutation effectively scrambles the resulting ciphertexts but not the randomness, thereby hindering an adversary (or a challenger) who tries to build an efficient recovery algorithm $\mathtt{Rec}$ for the transformed scheme. At the same time, we show how such construction is efficiently isometric, by showing an efficient circuit for the canonical type-2 encryption operator $U^{(2)}_{\mathtt{Enc}}$. This is formalised in the theorem below.

$\mathtt{KGen}()$	$\mathtt{Enc}_{pk}(m;r)$	$\mathtt{Dec}_{sk}(c)$
$(pk_e, sk_e) \leftarrow\!\!\$\ \mathtt{KGen}^{\Sigma}()$	**parse** pk **as** (pk_e, pk_f)	**parse** sk **as** (sk_e, sk_f)
$(pk_f, sk_f) \leftarrow\!\!\$\ \mathtt{KGen}^{F}()$	$y \leftarrow \mathtt{F}(pk_f, m)$	$y \leftarrow \mathtt{Dec}^{\Sigma}(sk_e, c)$
$pk \leftarrow (pk_e, pk_f)$	$c \leftarrow \mathtt{Enc}^{\Sigma}(pk_e, y; r)$	$m \leftarrow \mathtt{F}^{-1}(sk_f, y)$
$sk \leftarrow (sk_e, sk_f)$	**return** c	**return** m
return (pk, sk)		

Figure 6.7: Transformed scheme Pke, where $\Sigma = (\mathtt{KGen}^{\Sigma}, \mathtt{Enc}^{\Sigma}, \mathtt{Dec}^{\Sigma})$ is a public key encryption scheme and $\Pi = (\mathtt{KGen}^{F}, \mathtt{F}, \mathtt{F}^{-1})$ is a deterministic trapdoor permutation.

Theorem 6.3.4. *Let $\Pi = (\mathtt{KGen}^{F}, \mathtt{F}, \mathtt{F}^{-1})$ be a deterministic trapdoor permutation and $\Sigma = (\mathtt{KGen}^{\Sigma}, \mathtt{Enc}^{\Sigma}, \mathtt{Dec}^{\Sigma})$ be a PKE scheme. Let further* Pke $= (\mathtt{KGen}, \mathtt{Enc}, \mathtt{Dec})$ *be the PKE scheme constructed from Π and Σ according to the transformation depicted in Figure 6.7. If Π is quantum-secure and Σ is recoverable and $(1-\alpha)$-correct, then* Pke *is $(1-\alpha)$-correct, non-recoverable, and efficiently isometric.*

Proof. Partial correctness of the PKE scheme Pke is inherited from the partial correctness of the underlying PKE scheme Σ, as permuting the messages does not change the overall decryption failure probability.

Assume, for sake of contradiction, that Pke is recoverable. Then there exists an efficient algorithm $\mathtt{Rec}$ that, on input pk, r, and $\mathtt{Enc}_{pk}(m;r)$, outputs m. We construct the following adversary $\mathcal{B}$ against the trapdoor permutation Π. It receives a public key pk_f

for the trapdoor permutation $\mathtt{F}$ along with $y \leftarrow \mathtt{F}_{pk_f}(x)$ for a random x, and is asked to find x. Adversary $\mathcal{B}$ computes $(pk_e, sk_e) \leftarrow\!\!\$\ \mathtt{KGen}^{\Sigma}()$, chooses $r \leftarrow\!\!\$\ \mathcal{R}$, computes $c \leftarrow \mathtt{Enc}^{\Sigma}_{pk_e}(y; r)$, and uses $pk = (pk_e, pk_f), r, c$ as an input to $\mathtt{Rec}$. By construction, we have $c = \mathtt{Enc}^{\Sigma}_{pk_e}(\mathtt{F}_{pk_f}(x); r) = \mathtt{Enc}_{pk}(x; r)$, hence $\mathtt{Rec}$ outputs x. So $\mathcal{B}$ can find the correct preimage with probability 1, hence, breaking the security of the trapdoor permutation. This contradicts the recoverability of PKE.

Finally, in Figure 6.8 we show an efficient circuit for the realisation of $U^{(2)}_{\mathtt{Enc}}$. This uses subcircuits for computing the type-1 operator for the trapdoor permutation and its inverse, given both the public key and secret key of the trapdoor permutation, and type-1 encryption operator and type-1 recovery operator for the underlying PKE scheme. □

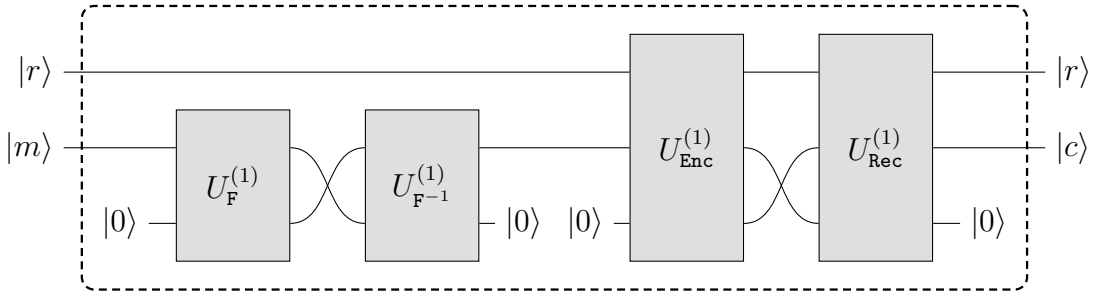

Figure 6.8: Efficient realisation of the canonical type-2 encryption operator for the construction shown in Figure 6.7.

Note that, albeit the above construction works at a theoretical level, there are currently no known candidates for quantum-secure trapdoor permutations. Alternatively, a quantum-secure injective trapdoor function could be used instead, for which candidates exist. In this case, because of the inherent expansion factor, the message space for the transformed scheme will be smaller than the one in the original public key encryption scheme.

6.4 Summary and Outlook

In this chapter, we have filled the existing gap between the symmetric key and the public key case when defining quantum security notions. We showed how the existence of this gap was not due to a mere lack of interest, but because of non-trivial definitional issues that we solved. Clearly, our notions of qIND-qCPA for PKE can be also used to study the security of cryptographic primitives that extend PKE with extra functionalities. Such primitives include, for example, fully homomorphic encryption [Gen09, BJ15], identity-based encryption [Zha12b], and functional encryption [BSW11]. Besides closing the aforementioned gap, our results also put forth several open questions that require further analysis.

An obvious challenge is the extension of our qIND-qCPA security to chosen ciphertext attacks. Chosen ciphertext attacks can be divided into the non-adaptive case [NY90]

and the adaptive case [RS92], whereas the formal distinction stems from [BDPR98]. The non-adaptive case (qIND-qCCA1) is easy as the adversary has full superposition access to the decryption oracle up to the point of making its challenge query, i.e., before learning the challenge ciphertext. The adaptive case (qIND-qCCA2), however, comes with the challenge of detecting a cheating adversary, i.e., one that simply tries to decrypt its challenge ciphertext. Trivial approaches are doomed to fail due to the destructive nature of measurements and the no-cloning theorem [WZ82, Die82]; while this challenge is closely related to the recording of superposition queries to a random oracle, we emphasise that the seminal work by Zhandry [Zha19a] does not apply. The reason is that the compressed oracle technique [Zha19a] works for random oracles which are modelled using type-1 operators, while our notion is based on type-2 operators. Instead, the technique presented in [AGM18], although falling into the realm of quantum encryption schemes, shows how to overcome this difficulty by using a real-vs-ideal approach. This makes it possible to differentiate the behaviour of the adversary when replaying the challenge ciphertext to the decryption oracle, hence effectively detecting a challenge replay attack. The technique provides a starting point towards a qIND-qCCA2 security notion.

Another open question regards the classification. We are unaware of a natural example for a PKE scheme that is isometric, yet not efficiently so. A simple idea would be to modify the construction from Figure 6.7 in such a way that the circuit provided in Figure 6.8 becomes non-efficient, for example by using a hard to invert permutation instead of a trapdoor permutation. However, this idea does not work for two reasons. First, it would only show that this particular construction of the type-2 operator is inefficient, while we would need to show that any construction is. Second, and more importantly, switching to a hard to invert permutation would make the decryption algorithm inefficient. Hence the resulting scheme would no longer be a PKE scheme according to Definition 2.2.7.

Extension to allowing randomness and public keys in superposition are further directions to be studied. We discussed two approaches for the former in Section 6.1.5 and showed inherent limitations. The question is whether there are meaningful notions between these approaches that yield stronger, yet achievable notions. While it is questionable whether notions allowing public keys in superposition are meaningful, there is an important observation we want to make here. It was shown that the LWE problem is easy when having access to a superposition of samples (using the same secret) [GKZ19]. Hence, for the canonical LWE-based PKE scheme, allowing a superposition of public keys would come with the inherent limitation that an adversary might simply obtain the secret key from the superposition of public keys it initially receives.

Finally, the exact relation between the qIND-qCPA security notion developed in this chapter and the alternative one proposed in [CEV20] is an open question. Incomparability for the symmetric encryption counterparts of the notions has been shown in [CETU21] which seems to extend via the standard hybrid encryption scheme also to the case of public key encryption. A classification of schemes for which the security notions are equivalent is an interesting question as it allows to analyse quantum security with respect to both notions by a single proof.

CHAPTER 7

CONCLUSION AND FUTURE DIRECTIONS

We studied the security of cryptographic primitives with respect to several advanced security notions, ranging from side-channel leakage over related-key attacks and related-randomness attacks to security against quantum adversaries. All these security notions cover attacks which are not covered by standard security notions. In Part I we developed the generic FGHF′ construction as well as the concrete sponge-based leakage-resilient AEAD scheme Slae. We further developed and analysed security notions considering related-key attacks and related-randomness attacks. In Part II, we showed the post-quantum security of several cryptographic primitives. Additionally, we developed a new quantum security notion that allows for a quantum indistinguishability phase.

We gave both positive result—like the leakage-resilient AEAD scheme Slae—as well as negative results—like showing many schemes insecure against resetting attacks and the insecurity of ROLLO-II with respect to our new quantum security notion. Hence, this thesis shows that careful analysis is required when considering advanced attacks. Each of the advanced security notions comes with further open questions that we described in the respective chapter. For this conclusion we take a broader view across all the notions.

We analysed each of the different advanced security notions individually. Recall the argument that advanced security notions are necessary to cover more elaborate attacks not covered by standard security notions, such as IND-CPA or IND-CCA. By the same argument, one can consider a combination of the advanced security notions to cover different advanced attacks simultaneously. In the following we want to motivate and make some observations on this.

We start with a combination of the notions considered in Part I by combining the leakage-resilient security notions from Chapter 3 and the related-key attack security notions from Chapter 4. Consider a device A which implements Slae and that comes with a hardened implementation that ensures a good bound on the number of bits that an adversary can obtain via leakage. The results from Chapter 3 provide concrete bounds

for the security of the scheme. Assuming that there is another device B with a related key but this implementation is not hardened with respect to side-channel leakage, i.e., the adversary can obtain much more information on the key of device B than on the key of device A. Depending on the relation between the keys and the exact leakage obtained from device B, the adversary might be able to get more leakage about device A than from the device itself due to the hardened implementation. This would effectively bypass the security guarantees from the leakage-resilient security proof via a side-channel attack on the related device.

We have scrutinised the sponge-based construction Slae and its underlying components SlFunc, SPrg, and SvHash with respect to side-channel leakage in Chapter 3 and in the QROM in Chapter 5. An obvious question is the simultaneous consideration of the QROM and side-channel leakage, thereby combining Part I and Part II. However, following [SPY+10], the idealised primitive is considered not to leak to prevent artificial attacks. In this sense, the combined notion would not be too different, except for the (leakage-free) superposition access to the underlying transformation ρ. However, the proof technique needs to be revisited as the leakage-resilient proofs for SlFunc rely on looking up queries that the adversary makes to ρ which does work in the QROM without modification.

The related-key attack and related-randomness attack security notions can be easily extended to the QROM to cover adversaries with local quantum computing power. Hence, the main question is whether existing classical proofs in the ROM (with respect to related-key attacks or related-randomness attacks) can be translated to post-quantum proofs in the QROM by revisiting the used proof techniques and adapting them to a corresponding QROM technique if necessary.

Considering side-channel leakage, as done in Chapter 3, in the setting of quantum security, considered in Chapter 6 is more involved. First, note that Slae, and in fact most AEAD schemes, are insecure with respect to quantum security notions that allow the challenge queries to be quantum [GHS16, MS16][38]. This effectively stems from the insecurity of the one-time pad against superposition attacks, similar to the attack we presented against ROLLO-II. A combination with side-channel attacks is meaningless if the schemes are already insecure even in the absence of side-channel leakage. Instead, one can consider the weaker notion IND-qCPA [BZ13b], where the learning queries are quantum while the challenge remains classical.[39] In combination with the leakage setting, this means that the adversary has classical, leakage-free oracles in the challenge phase and quantum oracles in the learning phase that additionally leak. The core question is what leakage on a quantum oracle is. Side-channel information is generally some classical information such as power consumption or runtime from which secret bits can be extracted. It is not clear what the power consumption of a computation in superposition actually tells the adversary. A related, yet different question are fault attacks [Krä15]. Here, the adversary injects a fault into the computation and derives some information based on the result of the (modified) computation. One can consider adversaries that inject faults into a superposition, which might yield more information than the classical counterpart.

[38] Note that we do not consider the security notion from Chapter 6 which is for public key encryption as Slae is a symmetric primitive.

[39] For this notion, secure constructions exist [ATTU16, BBC+21].

Another possible combination are related-randomness attacks in a quantum setting. In light of the results presented here, we focus again on the special case of resetting attacks, where random coins used for encryption are reused. Enhancing the qIND-qCPA notion[40] by resetting attacks comes with the same problem that arises when extending quantum security notions to the CCA scenario: detecting forbidden queries. In the CCA case, the challenge would need to copy the challenge ciphertext in order to prevent the adversary from simply asking for a decryption of the challenge ciphertext. The resetting attack security notions exclude forbidden queries via considering equality-pattern respecting adversaries. Recall that, for LoR-IND-CPRA, an adversary would violate the equality-pattern property if it makes a query (m_0, m_1) followed by querying (m_0, m_2). In general, this property has to be checked by the challenger; in the classical setting, this is trivial and one can simply assume the adversary to be equality-pattern respecting.[41] In the quantum setting, this is not the case as the challenger would need to copy the messages queried by the adversary which are arbitrary quantum states and hence thwarted by the no-cloning theorem. Note that the same also applies to the real-or-random notion RoR-IND-CPRA, where equality-pattern respecting means that the adversary does not repeat a query.

The QROM is implicitly considered in our quantum security qIND-qCPA. But note that the challenger still only accesses the random oracle classical. This stems from the fact that the (classical) randomness is input to the random oracle while the (quantum) message is not. In this sense, there is no difference to the established QROM from the post-quantum setting where only the adversary has quantum access to the random oracle. When considering the stronger setting of randomnesses in superposition, however, things change as the challenger would access the random oracle in superposition while generating the challenge ciphertext.

[40] Note that the same argument applies to the alternative quantum security notion given in [CEV20].

[41] This also appears in [BHK15] which study different flavours of handling forbidden queries to the decryption oracle in the classical setting.

LIST OF FIGURES

2.1 Security game SUFCMA. 9
2.2 Security game INDCPA. 10
2.3 Security game AE. 12
2.4 Security game INDCPA. 12
2.5 Security game INTCTXT. 12
2.6 Visualisation of the AEAD scheme N1 [NRS14]. 14
2.7 Visualisation of the AEAD scheme N2 [NRS14]. 14
2.8 Visualisation of the AEAD scheme N3 [NRS14]. 14
2.9 Pseudocode of N1, N2, and N3 in terms of a symmetric encryption scheme and a MAC. 15
2.10 Security game INDCPA. 16
2.11 Security game EUFCMA. 17
2.12 Security game EUFCMAHRK. 19
2.13 Security game PRF. 19
2.14 Security game PRG. 20
2.15 Plain sponge construction with four rounds of absorbing and two rounds of squeezing. 21
2.16 Security game LPRF. 23
2.17 Security game INDCPLA. 24
2.18 Security game INDaCPLA. 24
2.19 Security game SUFCMLA. 25
2.20 Security game LAE. 26
2.21 Security game SUFCMRKA. 28
2.22 Security games rkaPRF. 28
2.23 Security game lrINDCPRA using different randomnesses. 29
2.24 Security game lrINDCPRA. 30
2.25 Type-1 operator and type-2 operator for $\mathcal{F}$ 31
2.26 Security game INDqCPA. 34
2.27 Security game qIND. 34

2.28 Pseudocode of the canonical LWE-based public key encryption scheme. 36
2.29 Pseudocode of the lattice-based public key encryption scheme LARA. . 36
2.30 Pseudocode of the code-based public key encryption scheme ROLLO-II. 38
2.31 Pseudocode of the code-based public key encryption scheme HQC. . . . 38
2.32 Pseudocode of the code-based public key encryption scheme RQC. . . . 39
2.33 Hybrid encryption scheme. 39

3.1 Graphical representation of the FGHF′ construction. 47
3.2 Visualisation of the FGHF′ composition theorem (cf. Theorem 3.1.7). . 48
3.3 Pseudocode of the FGHF′ construction. 48
3.4 Games G_0, G_1, G_2, and G_3 used in the proof of Theorem 3.1.1. 51
3.5 Hybrid games H_i used in the proof of Theorem 3.1.1. 51
3.6 Security game LUF. 52
3.7 Games used in the proof of Theorem 3.1.4. 55
3.8 Games used in the proof of Theorem 3.1.6. 58
3.9 Pseudocode of the AEAD scheme SLAE. 60
3.10 Pseudocode of the sponge-based primitives SLFUNC, SPRG, and SVHASH. 61
3.11 Visualisation of the sponge-based function SLFUNC. 61
3.12 Visualisation of the sponge-based pseudorandom generator SPRG. . . . 61
3.13 Visualisation of the sponge-based hash function SVHASH. 62
3.14 Game G_0 used in the proof of Theorem 3.2.1. 68
3.15 Games G_1 and $\boxed{\mathsf{G}_1}$ used in the proof of Theorem 3.2.1. 68
3.16 Games G_2 and $\boxed{\mathsf{G}_2}$ used in the proof of Theorem 3.2.1. 69
3.17 Game G used in the proof of Theorem 3.2.2. 74
3.18 Graphical illustration of the hybrid game H_j used in the proof of Theorem 3.2.3. 75
3.19 Hybrid games H_j and $\boxed{\mathsf{H}_j}$ used in the proof of Theorem 3.2.3. 78
3.20 Games G_0 and $\boxed{\mathsf{G}_0}$ used in the proof of Theorem 3.2.3. 78
3.21 The sponge-based vector hash function SVHASH viewed as the standard sponge-based hash function SHASH with its inputs encoded via isPad. . 79
3.22 The padding schemes isPad and $\mathsf{lpad}^\star$. 80
3.23 Games G_0, G_1, and G_2 used in the proof of Lemma 3.3.4. 88
3.24 Hybrid games H_i used in the proof of Lemma 3.3.4. 88
3.25 Security game $\mathsf{SUFCMLA}_c$. 90
3.26 Games $\mathsf{G}_0, \mathsf{G}_1, \mathsf{G}_2, \mathsf{G}_3$ used in the proof of Theorem 3.3.6. 94
3.27 Games G_0, G_1, G_2, and G_3 used in the proof of Theorem 3.3.7. 98
3.28 Hybrid games H_i used in the proof of Theorem 3.3.7. 98
3.29 Games $\mathsf{G}_0, \mathsf{G}_1, \mathsf{G}_2$ used in the proof of Theorem 3.3.8. 100
3.30 Visualisation of the FGHF′ composition theorem (cf. Theorem 3.3.9). . 101

4.1 Security game s-rka-AE. 108
4.2 Security game rka-AE. 110
4.3 Security game INDCPRKA. 110
4.4 Games G_i used to prove Theorem 4.2.1 (RKA-AE security of N1). . . . 113
4.5 Games G_i used to prove Theorem 4.2.3 (RKA-AE security of N2). . . . 117

4.6 Games G_i used to prove Theorem 4.2.5 (RKA-AE security of N3). . . . 121
4.7 Visualisation of the encryption scheme $\textsc{Se}[\mathcal{F}, \mathcal{G}]$. 122
4.8 Games G_0, G_1, and G_2 used to prove the INDCPRKA security of Theorem 4.2.7. 124
4.9 Hybrid games H_i used to prove the INDCPRKA security of Theorem 4.2.7. 124
4.10 Visualisation of the message authentication code $\textsc{Mac}[\mathcal{H}, \mathcal{F}']$. 125
4.11 Games used in the proof of Theorem 4.2.8. 126
4.12 Games used in the proof of Theorem 4.2.9. 129
4.13 Separation example given in [Yil10b]. 129
4.14 LWE-based PKE scheme written as a PK-splittable scheme. 132
4.15 Code-based PKE scheme HQC written as a PK-splittable scheme. . . . 133
4.16 Code-based PKE scheme RQC written as a PK-splittable scheme. . . . 133
4.17 Code-based PKE scheme ROLLO-II written as a PK-splittable scheme. 134
4.18 Hybrid encryption scheme written as a PK-splittable scheme. 135
4.19 Hybrid games in the proof in [Yil10b]. 138
4.20 Hybrid games L_i and R_i used to prove Theorem 4.4.2. 141
4.21 Security game to define RoR-IND-CPRA security. 141
4.22 Real-or-random security dealing with repeating queries. 142
4.23 Hybrid games used to prove Theorem 4.4.5. 144
4.24 Games used to prove Lemma 4.4.7. 146

5.1 Security games INDCPA and pqINDCPA restricted to a single challenge for an oracle-simple PKE scheme $\textsc{Pke}$ using a random oracle H. 156
5.2 Algorithm Enc of an oracle-simple encryption scheme using F and Enc-Sub. 157
5.3 Security games INDCPA and pqINDCPA for an oracle-simple public key encryption scheme 158
5.4 Encryption scheme ROLLO-II written as oracle-simple encryption scheme. 161
5.5 Encryption scheme LARA written as an oracle-simple encryption scheme. 162
5.6 Visualisation of the sponge-based function $\textsc{SlFunc}$. 166
5.7 Visualisation of the sponge-based pseudorandom generator $\textsc{SPrg}$. . . . 167
5.8 Visualisation of the sponge-based hash function $\textsc{SvHash}$. 170
5.9 Generic construction for a deterministic wallet from a signature scheme with rerandomizable keys RSig and a random oracle H. 173
5.10 Security game WUNL. 174
5.11 Security game WUNF. 174
5.12 Game pq2Enc to define post-quantum security under double encryption. 183
5.13 Game G used in the proof of Theorem 5.4.5. 186

6.1 Canonical type-2 encryption operator for perfectly correct PKE schemes. 198
6.2 Canonical type-2 encryption operator for recoverable PKE schemes. . . 200
6.3 Security game qINDqCPA. 201
6.4 Generic attack against superposition of randomness. 203
6.5 Games $\mathsf{G}_{0,0}$, $\mathsf{G}_{0,1}$, $\mathsf{G}_{1,0}$, and $\mathsf{G}_{1,1}$ used in the proof of Theorem 6.2.6. . . 210
6.6 Classification of PKE schemes with respect to the qIND-qCPA security notion. 214

6.7 Transformed scheme PKE, where $\Sigma = (\mathtt{KGen}^{\Sigma}, \mathtt{Enc}^{\Sigma}, \mathtt{Dec}^{\Sigma})$ is a public key encryption scheme and $\Pi = (\mathtt{KGen}^{F}, \mathtt{F}, \mathtt{F}^{-1})$ is a deterministic trapdoor permutation. 215
6.8 Efficient realisation of the canonical type-2 encryption operator for the construction shown in Figure 6.7. 216

LIST OF ACRONYMS

AE authenticated encryption

AEAD authenticated encryption with associated data

CCA chosen ciphertext attacks

CCA1 non-adaptive chosen ciphertext attacks

CCA2 adaptive chosen ciphertext attacks

CPA chosen plaintext attacks

DLWE decisional learning with errors

DPA differential power analysis

E&M encrypt-and-MAC

EtM encrypt-then-MAC

EUF-CMA existential unforgeability under chosen message with attacks

EUF-CMA-HRK existential unforgeability under chosen message with attacks with honestly randomizable keys

FO Fujisaki-Okamoto

fqIND-CPA fully-quantum indistinguishability under chosen plaintext attacks

GC garbled circuits

HMAC hash-based message authentication code

ILRPC ideal low rank parity check

IND indistinguishability

IND-aCPLA indistinguishability under augmented chosen plaintext with leakage attacks

IND-CCA indistinguishability under chosen ciphertext attacks

IND-CCA1 indistinguishability under non-adaptive chosen ciphertext attacks

IND-CCA2 indistinguishability under adaptive chosen ciphertext attacks

IND-CPA indistinguishability under chosen plaintext attacks

IND-CPLA indistinguishability under chosen plaintext with leakage attacks

IND-CPRA indistinguishability under chosen plaintext and resetting attacks

IND-CPRKA indistinguishability under chosen plaintext and related-key attacks

IND-CPRRA indistinguishability under chosen plaintext and related-randomness attacks

IND-qCPA indistinguishability under quantum chosen plaintext attacks

INT-CTXT integrity of ciphertexts

INT-PTXT integrity of plaintexts

IRSR ideal rank support recovery

KEM key encapsulation mechanism

LAE leakage-resilient authenticated encryption

lbLWE low-bit learning with errors

LoR left-or-right

LPRF leakage-resilient pseudorandom function

LUF leakage-resilient unpredictable function

LWE learning with errors

MAC message authentication code

MPC multi-party computation

MtE MAC-then-encrypt

NIST National Institute of Standards and Technology

O2H one-way to hiding

OT oblivious transfer

OTP one-time pad

PKE public key encryption

PRF pseudorandom function

PRG pseudorandom generator

PRP pseudorandom permutation

qIND quantum indistinguishability

qIND-qCCA quantum indistinguishability under quantum chosen ciphertext attacks

qIND-qCCA1 quantum indistinguishability under non-adaptive quantum chosen ciphertext attacks

qIND-qCCA2 quantum indistinguishability under adaptive quantum chosen ciphertext attacks

qIND-qCPA quantum indistinguishability under quantum chosen plaintext attacks

qPRF quantum-secure pseudorandom function

QROM quantum random oracle model

RA resetting attacks

RKA related-key attacks

rkaAE related-key attack secure authenticated encryption

rkaPRF related-key attack secure pseudorandom function

ROM random oracle model

RoP real-or-permute

RoR real-or-random

RRA related-randomness attacks

SE symmetric encryption

SKE symmetric key encryption

SPA simple power analysis

srkaAE strong related-key attack secure authenticated encryption

STPC secure two-party computation

SUF-CMA strong unforgeability under chosen message attacks

SUF-CMLA strong unforgeability under chosen message with leakage attacks

SUF-CMRKA strong unforgeability under chosen message and related-key attacks

W-UNF wallet-unforgeability

W-UNL wallet-unlinkability

LIST OF NOTATION

$\mathbf{Adv}^{\mathsf{G}}_{\Sigma}(\mathcal{A})$	Advantage of adversary $\mathcal{A}$ when playing game G instantiated with primitive Σ
$\mathbf{Adv}^{\mathsf{G}}(\mathcal{A})$	Advantage of adversary $\mathcal{A}$ when playing game G
$\mathsf{G}^{\mathcal{A}} \Rightarrow x$	Game G outputs x when played by adversary $\mathcal{A}$
$\mathcal{A}^{\mathsf{G}} \Rightarrow x$	Adversary $\mathcal{A}$ outputs x when playing game G
$\leftarrow$	Deterministic assignment
$\leftarrow_{\$}$	Randomised assignment
$\mathcal{S} \leftarrow_{\cup} s$	Add s to set $\mathcal{S}$
$\mathcal{Y}^{\mathcal{X}}$	Set of functions from $\mathcal{X}$ to $\mathcal{Y}$
$\mathsf{Func}(\mathcal{X}, \mathcal{Y})$	Set of functions from $\mathcal{X}$ to $\mathcal{Y}$
$\mathsf{Func}(\mathcal{K}, \mathcal{X}, \mathcal{Y})$	Set of keyed functions from $\mathcal{X}$ to $\mathcal{Y}$ with key space $\mathcal{K}$
$[k]$	Set of integers from 1 to k
ε	Empty string
$\lceil X \rceil^{y}$	Leftmost y bits of X
$\lfloor X \rfloor_{y}$	Rightmost y bits of X
$\{0,1\}^{y}$	Set of bit string of length y
$\lvert \cdot \rangle$	Ket vector
$\langle \cdot \rvert$	Bra vector
$\mathbb{E}$	Expected value

REFERENCES

[AAB+19a] Carlos Aguilar Melchor, Nicolas Aragon, Slim Bettaieb, Loïc Bidoux, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, Edoardo Persichetti, and Gilles Zémor. HQC. Technical report, National Institute of Standards and Technology, 2019. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions. (Cited on pages 37, 132, and 206.)

[AAB+19b] Carlos Aguilar Melchor, Nicolas Aragon, Slim Bettaieb, Loic Bidoux, Olivier Blazy, Jean-Christophe Deneuville, Phillippe Gaborit, Gilles Zémor, Alain Couvreur, and Adrien Hauteville. RQC. Technical report, National Institute of Standards and Technology, 2019. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions. (Cited on pages 37, 39, 132, and 206.)

[AAG+16] Divesh Aggarwal, Shashank Agrawal, Divya Gupta, Hemanta K. Maji, Omkant Pandey, and Manoj Prabhakaran. Optimal computational split-state non-malleable codes. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A, Part II*, volume 9563 of *LNCS*, pages 393–417. Springer, Heidelberg, January 2016. doi:10.1007/978-3-662-49099-0_15. (Cited on page 109.)

[ABB+17] Erdem Alkim, Nina Bindel, Johannes A. Buchmann, Özgür Dagdelen, Edward Eaton, Gus Gutoski, Juliane Krämer, and Filip Pawlega. Revisiting TESLA in the quantum random oracle model. In Tanja Lange and Tsuyoshi Takagi, editors, *Post-Quantum Cryptography - 8th International Workshop, PQCrypto 2017*, pages 143–162. Springer, Heidelberg, 2017. doi:10.1007/978-3-319-59879-6_9. (Cited on page 153.)

[ABC+18] Megha Agrawal, Tarun Kumar Bansal, Donghoon Chang, Amit Kumar Chauhan, Seokhie Hong, Jinkeon Kang, and Somitra Kumar Sanadhya. RCB: leakage-resilient authenticated encryption via re-keying. *J. Supercomput.*, 74(9):4173–4198, 2018. doi:10.1007/s11227-016-1824-6. (Cited on page 45.)

[ABD+17] Nicolas Aragon, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, Adrien Hauteville, Olivier Ruatta, Jean-Pierre Tillich, and Gilles Zémor. LOCKER. Technical report, National Institute of Standards and Technology, 2017. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions. (Cited on page 157.)

[ABD+19] Nicolas Aragon, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, Adrien Hauteville, Olivier Ruatta, Jean-Pierre Tillich, Gilles Zémor, Carlos Aguilar Melchor, Slim Bettaieb, Loic Bidoux, Magali Bardet, and Ayoub Otmani. ROLLO. Technical report, National Institute of Standards and Technology, 2019. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions. (Cited on pages 36, 37, 132, 154, 161, 162, 192, and 206.)

[ABF13] Michel Abdalla, Sonia Belaïd, and Pierre-Alain Fouque. Leakage-resilient symmetric encryption via re-keying. In Guido Bertoni and Jean-Sébastien Coron, editors, *CHES 2013*, volume 8086 of *LNCS*, pages 471–488. Springer, Heidelberg, August 2013. doi:10.1007/978-3-642-40349-1_27. (Cited on page 45.)

[ABF+16] Gorjan Alagic, Anne Broadbent, Bill Fefferman, Tommaso Gagliardoni, Christian Schaffner, and Michael St. Jules. Computational security of quantum encryption. In Anderson C. A. Nascimento and Paulo Barreto, editors, *ICITS 16*, volume 10015 of *LNCS*, pages 47–71. Springer, Heidelberg, August 2016. doi:10.1007/978-3-319-49175-2_3. (Cited on page 193.)

[ABKM21] Gorjan Alagic, Chen Bai, Jonathan Katz, and Christian Majenz. Post-quantum security of the even-mansour cipher. *IACR Cryptol. ePrint Arch.*, page 1601, 2021. URL: https://eprint.iacr.org/2021/1601. (Cited on page 154.)

[ABL17] Farzaneh Abed, Francesco Berti, and Stefan Lucks. Is RCB a leakage resilient authenticated encryption scheme? In Helger Lipmaa, Aikaterini Mitrokotsa, and Raimundas Matulevicius, editors, *Secure IT Systems - 22nd Nordic Conference, NordSec 2017, Tartu, Estonia, November 8-10, 2017, Proceedings*, volume 10674 of *Lecture Notes in Computer Science*, pages 39–52. Springer, 2017. doi:10.1007/978-3-319-70290-2_3. (Cited on page 45.)

[ABPP18] Michel Abdalla, Fabrice Benhamouda, Alain Passelègue, and Kenneth G. Paterson. Related-key security for pseudorandom functions beyond the linear barrier. *Journal of Cryptology*, 31(4):917–964, October 2018. doi:10.1007/s00145-017-9274-8. (Cited on pages 104 and 105.)

[AC12] Scott Aaronson and Paul Christiano. Quantum money from hidden subspaces. In Howard J. Karloff and Toniann Pitassi, editors, *44th ACM STOC*, pages 41–60. ACM Press, May 2012. doi:10.1145/2213977.2213983. (Cited on page 31.)

[ADE+20] Nabil Alkeilani Alkadri, Poulami Das, Andreas Erwig, Sebastian Faust, Juliane Krämer, Siavash Riahi, and Patrick Struck. Deterministic wallets in a quantum world. In Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *ACM CCS 2020*, pages 1017–1031. ACM Press, November 2020. doi:10.1145/3372297.3423361. (Cited on pages 5, 151, and 187.)

[ADL17] Tomer Ashur, Orr Dunkelman, and Atul Luykx. Boosting authenticated encryption robustness with minimal modifications. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part III*, volume 10403 of *LNCS*, pages 3–33. Springer, Heidelberg, August 2017. doi:10.1007/978-3-319-63697-9_1. (Cited on page 90.)

[ADPS16] Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. Post-quantum key exchange - A new hope. In Thorsten Holz and Stefan Savage, editors, *USENIX Security 2016*, pages 327–343. USENIX Association, August 2016. (Cited on page 35.)

[AFPW11] Martin R. Albrecht, Pooya Farshim, Kenneth G. Paterson, and Gaven J. Watson. On cipher-dependent related-key attacks in the ideal-cipher model. In Antoine Joux, editor, *FSE 2011*, volume 6733 of *LNCS*, pages 128–145. Springer, Heidelberg, February 2011. doi:10.1007/978-3-642-21702-9_8. (Cited on pages 28 and 105.)

[AGM18] Gorjan Alagic, Tommaso Gagliardoni, and Christian Majenz. Unforgeable quantum encryption. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part III*, volume 10822 of *LNCS*, pages 489–519. Springer, Heidelberg, April / May 2018. doi:10.1007/978-3-319-78372-7_16. (Cited on pages 193 and 217.)

[AGM21] Gorjan Alagic, Tommaso Gagliardoni, and Christian Majenz. Can you sign a quantum state? *Quantum*, 5:603, 2021. (Cited on page 193.)

[AHI11] Benny Applebaum, Danny Harnik, and Yuval Ishai. Semantic security under related-key attacks and applications. In *Innovations in Computer Science - ICS 2011, Tsinghua University, Beijing, China, January 7-9, 2011. Proceedings*. Tsinghua University Press, 2011. (Cited on pages 104 and 105.)

[AHU19] Andris Ambainis, Mike Hamburg, and Dominique Unruh. Quantum security proofs using semi-classical oracles. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part II*, volume 11693 of *LNCS*, pages 269–295. Springer, Heidelberg, August 2019. doi:10.1007/978-3-030-26951-7_10. (Cited on pages 32, 155, and 187.)

[Ajt96] Miklós Ajtai. Generating hard instances of lattice problems (extended abstract). In *28th ACM STOC*, pages 99–108. ACM Press, May 1996. doi:10.1145/237814.237838. (Cited on page 152.)

[AMRS20] Gorjan Alagic, Christian Majenz, Alexander Russell, and Fang Song. Quantum-access-secure message authentication via blind-unforgeability. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part III*, volume 12107 of *LNCS*, pages 788–817. Springer, Heidelberg, May 2020. doi:10.1007/978-3-030-45727-3_27. (Cited on pages 191 and 193.)

[AOP+17] Martin R. Albrecht, Emmanuela Orsini, Kenneth G. Paterson, Guy Peer, and Nigel P. Smart. Tightly secure ring-LWE based key encapsulation with short ciphertexts. In Simon N. Foley, Dieter Gollmann, and Einar Snekkenes, editors, *ESORICS 2017, Part I*, volume 10492 of *LNCS*, pages 29–46. Springer, Heidelberg, September 2017. doi:10.1007/978-3-319-66402-6_4. (Cited on pages 35 and 157.)

[AR17] Gorjan Alagic and Alexander Russell. Quantum-secure symmetric-key cryptography based on hidden shifts. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part III*, volume 10212 of *LNCS*, pages 65–93. Springer, Heidelberg, April / May 2017. doi:10.1007/978-3-319-56617-7_3. (Cited on pages 190 and 193.)

[ARU14] Andris Ambainis, Ansis Rosmanis, and Dominique Unruh. Quantum attacks on classical proof systems: The hardness of quantum rewinding. In *55th FOCS*, pages 474–483. IEEE Computer Society Press, October 2014. doi:10.1109/FOCS.2014.57. (Cited on page 152.)

[ATTU16] Mayuresh Vivekanand Anand, Ehsan Ebrahimi Targhi, Gelo Noel Tabia, and Dominique Unruh. Post-quantum security of the CBC, CFB, OFB, CTR, and XTS modes of operation. In Tsuyoshi Takagi, editor, *Post-Quantum Cryptography - 7th International Workshop, PQCrypto 2016*, pages 44–63. Springer, Heidelberg, 2016. doi:10.1007/978-3-319-29360-8_4. (Cited on pages 193 and 220.)

[Ban19] Rachid El Bansarkhani. LARA: A design concept for lattice-based encryption. In Ian Goldberg and Tyler Moore, editors, *FC 2019*, volume 11598 of *LNCS*, pages 377–395. Springer, Heidelberg, February 2019. doi:10.1007/978-3-030-32101-7_23. (Cited on pages 36, 154, 157, 161, and 163.)

[BBB+20] Davide Bellizia, Francesco Berti, Olivier Bronchain, Gaëtan Cassiers, Sébastien Duval, Chun Guo, Gregor Leander, Gaëtan Leurent, Itamar Levi, Charles Momin, Olivier Pereira, Thomas Peters, François-Xavier Standaert, Balazs Udvarhelyi, and Friedrich Wiemer. Spook: Sponge-based leakage-resistant authenticated encryption with a masked tweakable block cipher. *IACR Trans. Symm. Cryptol.*, 2020(S1):295–349, 2020. doi:10.13154/tosc.v2020.iS1.295-349. (Cited on page 20.)

[BBC+20] Davide Bellizia, Olivier Bronchain, Gaëtan Cassiers, Vincent Grosso, Chun Guo, Charles Momin, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. Mode-level vs. implementation-level physical security in symmetric cryptography - A practical guide through the leakage-resistance jungle. In

Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part I*, volume 12170 of *LNCS*, pages 369–400. Springer, Heidelberg, August 2020. doi:10.1007/978-3-030-56784-2_13. (Cited on page 101.)

[BBC+21] Ritam Bhaumik, Xavier Bonnetain, André Chailloux, Gaëtan Leurent, María Naya-Plasencia, André Schrottenloher, and Yannick Seurin. QCB: efficient quantum-secure authenticated encryption. In Mehdi Tibouchi and Huaxiong Wang, editors, *Advances in Cryptology - ASIACRYPT 2021 - 27th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 6-10, 2021, Proceedings, Part I*, volume 13090 of *Lecture Notes in Computer Science*, pages 668–698. Springer, 2021. doi:10.1007/978-3-030-92062-3_23. (Cited on page 220.)

[BBD09] Daniel J Bernstein, Johannes Buchmann, and Erik Dahmen. *Post-Quantum Cryptography*. Springer-Verlag, Berlin-Heidleberg.–245 p, 2009. (Cited on page 151.)

[BBdS+20] Christof Beierle, Alex Biryukov, Luan Cardoso dos Santos, Johann Großschädl, Léo Perrin, Aleksei Udovenko, Vesselin Velichkov, Qingju Wang, Amir Moradi, and Aein Rezaei Shahmirzadi. Sparkle. Technical report, National Institute of Standards and Technology, 2020. available at https://csrc.nist.gov/projects/lightweight-cryptography/finalists. (Cited on page 154.)

[BBN+09] Mihir Bellare, Zvika Brakerski, Moni Naor, Thomas Ristenpart, Gil Segev, Hovav Shacham, and Scott Yilek. Hedged public-key encryption: How to protect against bad randomness. In Mitsuru Matsui, editor, *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 232–249. Springer, Heidelberg, December 2009. doi:10.1007/978-3-642-10366-7_14. (Cited on page 106.)

[BC10] Mihir Bellare and David Cash. Pseudorandom functions and permutations provably secure against related-key attacks. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 666–684. Springer, Heidelberg, August 2010. doi:10.1007/978-3-642-14623-7_36. (Cited on pages 28, 104, and 105.)

[BCD+20] Zhenzhen Bao, Avik Chakraborti, Nilanjan Datta, Jian Guo, Mridul Nandi, Thomas Peyrin, and Kan Yasuda. Photon-beetle. Technical report, National Institute of Standards and Technology, 2020. available at https://csrc.nist.gov/projects/lightweight-cryptography/finalists. (Cited on page 154.)

[BCDM20] Tim Beyne, Yu Long Chen, Christoph Dobraunig, and Bart Mennink. Elephant. Technical report, National Institute of Standards and Technology, 2020. available at https://csrc.nist.gov/projects/lightweight-cryptography/finalists. (Cited on page 154.)

[BCM11] Mihir Bellare, David Cash, and Rachel Miller. Cryptography secure against related-key attacks and tampering. In Dong Hoon Lee and Xiaoyun Wang, editors, *ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 486–503. Springer, Heidelberg, December 2011. doi:10.1007/978-3-642-25385-0_26. (Cited on pages 28 and 106.)

[BCM+18] Zvika Brakerski, Paul Christiano, Urmila Mahadev, Umesh V. Vazirani, and Thomas Vidick. A cryptographic test of quantumness and certifiable randomness from a single quantum device. In Mikkel Thorup, editor, *59th FOCS*, pages 320–331. IEEE Computer Society Press, October 2018. doi:10.1109/FOCS.2018.00038. (Cited on page 153.)

[BDF+11] Dan Boneh, Özgür Dagdelen, Marc Fischlin, Anja Lehmann, Christian Schaffner, and Mark Zhandry. Random oracles in a quantum world. In Dong Hoon Lee and Xiaoyun Wang, editors, *ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 41–69. Springer, Heidelberg, December 2011. doi:10.1007/978-3-642-25385-0_3. (Cited on pages 3, 32, 153, 154, 155, 190, and 193.)

[BDG20] Mihir Bellare, Hannah Davis, and Felix Günther. Separate your domains: NIST PQC KEMs, oracle cloning and read-only indifferentiability. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part II*, volume 12106 of *LNCS*, pages 3–32. Springer, Heidelberg, May 2020. doi:10.1007/978-3-030-45724-2_1. (Cited on page 198.)

[BDJR97] Mihir Bellare, Anand Desai, Eric Jokipii, and Phillip Rogaway. A concrete security treatment of symmetric encryption. In *38th FOCS*, pages 394–403. IEEE Computer Society Press, October 1997. doi:10.1109/SFCS.1997.646128. (Cited on page 10.)

[BDK05] Eli Biham, Orr Dunkelman, and Nathan Keller. Related-key boomerang and rectangle attacks. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 507–525. Springer, Heidelberg, May 2005. doi:10.1007/11426639_30. (Cited on page 104.)

[BDK+18] Joppe W. Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé. CRYSTALS - kyber: A cca-secure module-lattice-based KEM. In *2018 IEEE European Symposium on Security and Privacy, EuroS&P 2018, London, United Kingdom, April 24-26, 2018*, pages 353–367. IEEE, 2018. doi:10.1109/EuroSP.2018.00032. (Cited on page 35.)

[BDK+20] Niklas Büscher, Daniel Demmler, Nikolaos P. Karvelas, Stefan Katzenbeisser, Juliane Krämer, Deevashwer Rathee, Thomas Schneider, and Patrick Struck. Secure two-party computation in a quantum world. In Mauro Conti, Jianying Zhou, Emiliano Casalicchio, and Angelo Spognardi, editors, *ACNS 20, Part I*, volume 12146 of *LNCS*, pages 461–480. Springer, Heidelberg, October 2020. doi:10.1007/978-3-030-57808-4_23. (Cited on pages 6 and 151.)

[BDL97] Dan Boneh, Richard A. DeMillo, and Richard J. Lipton. On the importance of checking cryptographic protocols for faults (extended abstract). In Walter Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 37–51. Springer, Heidelberg, May 1997. doi:10.1007/3-540-69053-0_4. (Cited on page 43.)

[BDPA13] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Keccak. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 313–314. Springer, Heidelberg, May 2013. doi:10.1007/978-3-642-38348-9_19. (Cited on page 101.)

[BDPR98] Mihir Bellare, Anand Desai, David Pointcheval, and Phillip Rogaway. Relations among notions of security for public-key encryption schemes. In Hugo Krawczyk, editor, *CRYPTO'98*, volume 1462 of *LNCS*, pages 26–45. Springer, Heidelberg, August 1998. doi:10.1007/BFb0055718. (Cited on page 217.)

[BDPV08] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. On the indifferentiability of the sponge construction. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 181–197. Springer, Heidelberg, April 2008. doi:10.1007/978-3-540-78967-3_11. (Cited on page 20.)

[BDPV10] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Sponge-based pseudo-random number generators. In Stefan Mangard and François-Xavier Standaert, editors, *CHES 2010*, volume 6225 of *LNCS*, pages 33–47. Springer, Heidelberg, August 2010. doi:10.1007/978-3-642-15031-9_3. (Cited on pages 62 and 73.)

[BDPV12] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Duplexing the sponge: Single-pass authenticated encryption and other applications. In Ali Miri and Serge Vaudenay, editors, *SAC 2011*, volume 7118 of *LNCS*, pages 320–337. Springer, Heidelberg, August 2012. doi:10.1007/978-3-642-28496-0_19. (Cited on page 21.)

[BDPVA07] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Sponge functions. In *ECRYPT Hash Workshop*, 2007. https://keccak.team/files/SpongeFunctions.pdf. (Cited on pages 20, 62, 82, and 83.)

[Ber14] Daniel J. Bernstein. CAESAR: Competition for Authenticated Encryption: Security, Applicability, and Robustness, 2014. URL: http://competitions.cr.yp.to/. (Cited on pages 44 and 104.)

[BF15] Manuel Barbosa and Pooya Farshim. The related-key analysis of Feistel constructions. In Carlos Cid and Christian Rechberger, editors, *FSE 2014*, volume 8540 of *LNCS*, pages 265–284. Springer, Heidelberg, March 2015. doi:10.1007/978-3-662-46706-0_14. (Cited on pages 104 and 105.)

[BGP+19] Francesco Berti, Chun Guo, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. TEDT: a leakage-resistant AEAD mode. *IACR TCHES*,

2020(1):256–320, 2019. https://tches.iacr.org/index.php/TCHES/article/view/8400. doi:10.13154/tches.v2020.i1.256-320. (Cited on pages 44 and 45.)

[BGV12] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In Shafi Goldwasser, editor, *ITCS 2012*, pages 309–325. ACM, January 2012. doi:10.1145/2090236.2090262. (Cited on page 35.)

[BH15] Mihir Bellare and Viet Tung Hoang. Resisting randomness subversion: Fast deterministic and hedged public-key encryption in the standard model. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 627–656. Springer, Heidelberg, April 2015. doi:10.1007/978-3-662-46803-6_21. (Cited on page 106.)

[BHH+19] Nina Bindel, Mike Hamburg, Kathrin Hövelmanns, Andreas Hülsing, and Edoardo Persichetti. Tighter proofs of CCA security in the quantum random oracle model. In Dennis Hofheinz and Alon Rosen, editors, *TCC 2019, Part II*, volume 11892 of *LNCS*, pages 61–90. Springer, Heidelberg, December 2019. doi:10.1007/978-3-030-36033-7_3. (Cited on pages 32, 155, and 187.)

[BHK15] Mihir Bellare, Dennis Hofheinz, and Eike Kiltz. Subtleties in the definition of IND-CCA: When and how should challenge decryption be disallowed? *Journal of Cryptology*, 28(1):29–48, January 2015. doi:10.1007/s00145-013-9167-4. (Cited on pages 3 and 221.)

[BHN+19] Xavier Bonnetain, Akinori Hosoyamada, María Naya-Plasencia, Yu Sasaki, and André Schrottenloher. Quantum attacks without superposition queries: The offline Simon's algorithm. In Steven D. Galbraith and Shiho Moriai, editors, *ASIACRYPT 2019, Part I*, volume 11921 of *LNCS*, pages 552–583. Springer, Heidelberg, December 2019. doi:10.1007/978-3-030-34578-5_20. (Cited on page 193.)

[Bih94] Eli Biham. New types of cryptanalytic attacks using related keys (extended abstract). In Tor Helleseth, editor, *EUROCRYPT'93*, volume 765 of *LNCS*, pages 398–409. Springer, Heidelberg, May 1994. doi:10.1007/3-540-48285-7_34. (Cited on pages 104 and 105.)

[BJ15] Anne Broadbent and Stacey Jeffery. Quantum homomorphic encryption for circuits of low T-gate complexity. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 609–629. Springer, Heidelberg, August 2015. doi:10.1007/978-3-662-48000-7_30. (Cited on pages 193 and 216.)

[BK03] Mihir Bellare and Tadayoshi Kohno. A theoretical treatment of related-key attacks: RKA-PRPs, RKA-PRFs, and applications. In Eli Biham, editor,

EUROCRYPT 2003, volume 2656 of *LNCS*, pages 491–506. Springer, Heidelberg, May 2003. doi:10.1007/3-540-39200-9_31. (Cited on pages 27, 28, 104, and 105.)

[BK09] Alex Biryukov and Dmitry Khovratovich. Related-key cryptanalysis of the full AES-192 and AES-256. In Mitsuru Matsui, editor, *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 1–18. Springer, Heidelberg, December 2009. doi:10.1007/978-3-642-10366-7_1. (Cited on page 104.)

[BKN09] Alex Biryukov, Dmitry Khovratovich, and Ivica Nikolic. Distinguisher and related-key attack on the full AES-256. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 231–249. Springer, Heidelberg, August 2009. doi:10.1007/978-3-642-03356-8_14. (Cited on pages 104 and 106.)

[BKP+16] Francesco Berti, François Koeune, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. Leakage-resilient and misuse-resistant authenticated encryption. Cryptology ePrint Archive, Report 2016/996, 2016. https://eprint.iacr.org/2016/996. (Cited on page 45.)

[BKVV20] Zvika Brakerski, Venkata Koppula, Umesh V. Vazirani, and Thomas Vidick. Simpler proofs of quantumness. In Steven T. Flammia, editor, *15th Conference on the Theory of Quantum Computation, Communication and Cryptography, TQC 2020, June 9-12, 2020, Riga, Latvia*, volume 158 of *LIPIcs*, pages 8:1–8:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.TQC.2020.8. (Cited on page 153.)

[BL20] Anne Broadbent and Sébastien Lord. Uncloneable quantum encryption via oracles. In Steven T. Flammia, editor, *15th Conference on the Theory of Quantum Computation, Communication and Cryptography, TQC 2020, June 9-12, 2020, Riga, Latvia*, volume 158 of *LIPIcs*, pages 4:1–4:22. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.TQC.2020.4. (Cited on page 33.)

[BLO18] Carsten Baum, Huang Lin, and Sabine Oechsner. Towards practical lattice-based one-time linkable ring signatures. In David Naccache, Shouhuai Xu, Sihan Qing, Pierangela Samarati, Gregory Blanc, Rongxing Lu, Zonghua Zhang, and Ahmed Meddahi, editors, *ICICS 18*, volume 11149 of *LNCS*, pages 303–322. Springer, Heidelberg, October 2018. doi:10.1007/978-3-030-01950-1_18. (Cited on page 153.)

[BMOS17] Guy Barwell, Daniel P. Martin, Elisabeth Oswald, and Martijn Stam. Authenticated encryption in the face of protocol and side channel leakage. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT 2017, Part I*, volume 10624 of *LNCS*, pages 693–723. Springer, Heidelberg, December 2017. doi:10.1007/978-3-319-70694-8_24. (Cited on pages 22, 24, 26, 44, 45, 46, 47, 58, 84, 89, 90, 93, and 95.)

[BN00] Mihir Bellare and Chanathip Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In Tatsuaki Okamoto, editor, *ASIACRYPT 2000*, volume 1976 of *LNCS*, pages 531–545. Springer, Heidelberg, December 2000. doi:10.1007/3-540-44448-3_41. (Cited on pages 13, 44, and 111.)

[BN06] Mihir Bellare and Gregory Neven. Multi-signatures in the plain public-key model and a general forking lemma. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM CCS 2006*, pages 390–399. ACM Press, October / November 2006. doi:10.1145/1180405.1180453. (Cited on page 152.)

[BN08] Mihir Bellare and Chanathip Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. *Journal of Cryptology*, 21(4):469–491, October 2008. doi:10.1007/s00145-008-9026-x. (Cited on pages 13 and 44.)

[BNS19] Xavier Bonnetain, María Naya-Plasencia, and André Schrottenloher. On quantum slide attacks. In Kenneth G. Paterson and Douglas Stebila, editors, *SAC 2019*, volume 11959 of *LNCS*, pages 492–519. Springer, Heidelberg, August 2019. doi:10.1007/978-3-030-38471-5_20. (Cited on page 193.)

[BPP18] Francesco Berti, Olivier Pereira, and Thomas Peters. Reconsidering generic composition: The tag-then-encrypt case. In Debrup Chakraborty and Tetsu Iwata, editors, *INDOCRYPT 2018*, volume 11356 of *LNCS*, pages 70–90. Springer, Heidelberg, December 2018. doi:10.1007/978-3-030-05378-9_4. (Cited on page 13.)

[BPPS17] Francesco Berti, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. On leakage-resilient authenticated encryption with decryption leakages. *IACR Trans. Symm. Cryptol.*, 2017(3):271–293, 2017. doi:10.13154/tosc.v2017.i3.271-293. (Cited on pages 44 and 45.)

[BR93] 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 CCS 93*, pages 62–73. ACM Press, November 1993. doi:10.1145/168588.168596. (Cited on pages 3, 31, and 153.)

[BR06] Mihir Bellare and Phillip Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 409–426. Springer, Heidelberg, May / June 2006. doi:10.1007/11761679_25. (Cited on page 8.)

[BR14] Rishiraj Bhattacharyya and Arnab Roy. Secure message authentication against related-key attack. In Shiho Moriai, editor, *FSE 2013*, volume 8424 of *LNCS*, pages 305–324. Springer, Heidelberg, March 2014. doi:10.1007/978-3-662-43933-3_16. (Cited on pages 27, 104, and 105.)

[BS97] Eli Biham and Adi Shamir. Differential fault analysis of secret key cryptosystems. In Burton S. Kaliski Jr., editor, *CRYPTO'97*, volume 1294 of *LNCS*, pages 513–525. Springer, Heidelberg, August 1997. doi:10.1007/BFb0052259. (Cited on page 43.)

[BS20] Dan Boneh and Victor Shoup. *A Graduate Course in Applied Cryptography*. 2020. Draft 0.5. URL: http://toc.cryptobook.us/. (Cited on pages 79, 80, and 81.)

[BSW11] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In Yuval Ishai, editor, *TCC 2011*, volume 6597 of *LNCS*, pages 253–273. Springer, Heidelberg, March 2011. doi:10.1007/978-3-642-19571-6_16. (Cited on page 216.)

[BZ13a] Dan Boneh and Mark Zhandry. Quantum-secure message authentication codes. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 592–608. Springer, Heidelberg, May 2013. doi:10.1007/978-3-642-38348-9_35. (Cited on page 191.)

[BZ13b] Dan Boneh and Mark Zhandry. Secure signatures and chosen ciphertext security in a quantum computing world. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 361–379. Springer, Heidelberg, August 2013. doi:10.1007/978-3-642-40084-1_21. (Cited on pages 33, 190, 191, 193, 195, 201, and 220.)

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd FOCS*, pages 136–145. IEEE Computer Society Press, October 2001. doi:10.1109/SFCS.2001.959888. (Cited on page 2.)

[CBB+17] Alain Couvreur, Magali Bardet, Elise Barelli, Olivier Blazy, Rodolfo Canto-Torres, Philippe Gaborit, Ayoub Otmani, Nicolas Sendrier, and Jean-Pierre Tillich. BIG QUAKE. Technical report, National Institute of Standards and Technology, 2017. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions. (Cited on pages 157, 206, and 207.)

[CBH+17] Jan Czajkowski, Leon Groot Bruinderink, Andreas Hülsing, Christian Schaffner, and Dominique Unruh. Post-quantum security of the sponge construction. Cryptology ePrint Archive, Report 2017/771, 2017. https://eprint.iacr.org/2017/771. (Cited on page 170.)

[CBH+18] Jan Czajkowski, Leon Groot Bruinderink, Andreas Hülsing, Christian Schaffner, and Dominique Unruh. Post-quantum security of the sponge construction. In Tanja Lange and Rainer Steinwandt, editors, *Post-Quantum Cryptography - 9th International Conference, PQCrypto 2018*, pages 185–204. Springer, Heidelberg, 2018. doi:10.1007/978-3-319-79063-3_9. (Cited on pages 155 and 170.)

[CCL20] Nai-Hui Chia, Kai-Min Chung, and Ching-Yi Lai. On the need for large quantum depth. In Konstantin Makarychev, Yury Makarychev, Madhur Tulsiani, Gautam Kamath, and Julia Chuzhoy, editors, *52nd ACM STOC*, pages 902–915. ACM Press, June 2020. doi:10.1145/3357713.3384291. (Cited on page 33.)

[CCLY21] Nai-Hui Chia, Kai-Min Chung, Qipeng Liu, and Takashi Yamakawa. On the impossibility of post-quantum black-box zero-knowledge in constant rounds, 2021. arXiv:2103.11244. (Cited on page 33.)

[CEJv03] Stanley Chow, Philip A. Eisen, Harold Johnson, and Paul C. van Oorschot. White-box cryptography and an AES implementation. In Kaisa Nyberg and Howard M. Heys, editors, *SAC 2002*, volume 2595 of *LNCS*, pages 250–270. Springer, Heidelberg, August 2003. doi:10.1007/3-540-36492-7_17. (Cited on page 190.)

[CETU21] Tore Vincent Carstens, Ehsan Ebrahimi, Gelo Noel Tabia, and Dominique Unruh. Relationships between quantum IND-CPA notions. In Kobbi Nissim and Brent Waters, editors, *TCC 2021*, volume 13042 of *Lecture Notes in Computer Science*, pages 240–272. Springer, 2021. doi:10.1007/978-3-030-90459-3_9. (Cited on pages 193 and 217.)

[CEV20] Céline Chevalier, Ehsan Ebrahimi, and Quoc-Huy Vu. On security notions for encryption in a quantum world. Cryptology ePrint Archive, Report 2020/237, 2020. https://eprint.iacr.org/2020/237. (Cited on pages 191, 193, 213, 217, and 221.)

[Cha19] André Chailloux. Quantum security of the Fiat-Shamir transform of commit and open protocols. Cryptology ePrint Archive, Report 2019/699, 2019. https://eprint.iacr.org/2019/699. (Cited on page 155.)

[CJL+16] Lily Chen, Stephen Jordan, Yi-Kai Liu, Dustin Moody, Rene Peralta, Ray Perlner, and Daniel Smith-Tone. *Report on post-quantum cryptography*, volume 12. US Department of Commerce, National Institute of Standards and Technology, 2016. (Cited on page 152.)

[CJRR99] Suresh Chari, Charanjit S. Jutla, Josyula R. Rao, and Pankaj Rohatgi. Towards sound approaches to counteract power-analysis attacks. In Michael J. Wiener, editor, *CRYPTO'99*, volume 1666 of *LNCS*, pages 398–412. Springer, Heidelberg, August 1999. doi:10.1007/3-540-48405-1_26. (Cited on page 45.)

[CMP20] Andrea Coladangelo, Christian Majenz, and Alexander Poremba. Quantum copy-protection of compute-and-compare programs in the quantum random oracle model. Cryptology ePrint Archive, Report 2020/1194, 2020. https://eprint.iacr.org/2020/1194. (Cited on page 33.)

[CMSZ19] Jan Czajkowski, Christian Majenz, Christian Schaffner, and Sebastian Zur. Quantum lazy sampling and game-playing proofs for quantum indifferentiability. Cryptology ePrint Archive, Report 2019/428, 2019. https://eprint.iacr.org/2019/428. (Cited on page 33.)

[Cou06] Jean-Marc Couveignes. Hard homogeneous spaces. Cryptology ePrint Archive, Report 2006/291, 2006. https://eprint.iacr.org/2006/291. (Cited on page 152.)

[CRSS20] Valerio Cini, Sebastian Ramacher, Daniel Slamanig, and Christoph Striecks. CCA-secure (puncturable) KEMs from encryption with non-negligible decryption errors. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020, Part I*, volume 12491 of *LNCS*, pages 159–190. Springer, Heidelberg, December 2020. doi:10.1007/978-3-030-64837-4_6. (Cited on page 33.)

[Cza21] Jan Czajkowski. Quantum indifferentiability of SHA-3. *IACR Cryptol. ePrint Arch.*, 2021:192, 2021. URL: https://eprint.iacr.org/2021/192. (Cited on page 155.)

[DA99] Tim Dierks and Christopher Allen. *RFC 2246 - The TLS Protocol Version 1.0*. Internet Activities Board, January 1999. (Cited on page 106.)

[DDKA21] Mina Doosti, Mahshid Delavar, Elham Kashefi, and Myrto Arapinis. A unified framework for quantum unforgeability. *CoRR*, abs/2103.13994, 2021. URL: https://arxiv.org/abs/2103.13994, arXiv:2103.13994. (Cited on page 191.)

[DEM+17] Christoph Dobraunig, Maria Eichlseder, Stefan Mangard, Florian Mendel, and Thomas Unterluggauer. ISAP – towards side-channel secure authenticated encryption. *IACR Trans. Symm. Cryptol.*, 2017(1):80–105, 2017. doi:10.13154/tosc.v2017.i1.80-105. (Cited on pages 20, 44, 58, and 59.)

[DEM+20a] Christoph Dobraunig, Maria Eichlseder, Stefan Mangard, Florian Mendel, Bart Mennink, Robert Primas, and Thomas Unterluggauer. Isap. Technical report, National Institute of Standards and Technology, 2020. available at https://csrc.nist.gov/projects/lightweight-cryptography/finalists. (Cited on page 154.)

[DEM+20b] Christoph Dobraunig, Maria Eichlseder, Stefan Mangard, Florian Mendel, Bart Mennink, Robert Primas, and Thomas Unterluggauer. ISAP v2.0. *IACR Trans. Symm. Cryptol.*, 2020(S1):390–416, 2020. doi:10.13154/tosc.v2020.iS1.390-416. (Cited on pages 20, 44, 45, 58, and 59.)

[DEMS16] Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. Ascon v1.2. *Submission to the CAESAR Competition*, 2016. (Cited on pages 20 and 147.)

[DEMS20] Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. Ascon. Technical report, National Institute of Standards and Technology, 2020. available at https://csrc.nist.gov/projects/lightweight-cryptography/finalists. (Cited on page 154.)

[DF90] Yvo Desmedt and Yair Frankel. Threshold cryptosystems. In Gilles Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 307–315. Springer, Heidelberg, August 1990. doi:10.1007/0-387-34805-0_28. (Cited on page 187.)

[DFG13] Özgür Dagdelen, Marc Fischlin, and Tommaso Gagliardoni. The Fiat-Shamir transformation in a quantum world. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013, Part II*, volume 8270 of *LNCS*, pages 62–81. Springer, Heidelberg, December 2013. doi:10.1007/978-3-642-42045-0_4. (Cited on page 155.)

[DFL19] Poulami Das, Sebastian Faust, and Julian Loss. A formal treatment of deterministic wallets. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 651–668. ACM Press, November 2019. doi:10.1145/3319535.3354236. (Cited on pages 154, 171, 172, 175, 177, and 178.)

[DFM20] Jelle Don, Serge Fehr, and Christian Majenz. The measure-and-reprogram technique 2.0: Multi-round fiat-shamir and more. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part III*, volume 12172 of *LNCS*, pages 602–631. Springer, Heidelberg, August 2020. doi:10.1007/978-3-030-56877-1_21. (Cited on page 155.)

[DFMS19] Jelle Don, Serge Fehr, Christian Majenz, and Christian Schaffner. Security of the Fiat-Shamir transformation in the quantum random-oracle model. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part II*, volume 11693 of *LNCS*, pages 356–383. Springer, Heidelberg, August 2019. doi:10.1007/978-3-030-26951-7_13. (Cited on page 155.)

[DFMS21] Jelle Don, Serge Fehr, Christian Majenz, and Christian Schaffner. Online-extractability in the quantum random-oracle model. *IACR Cryptol. ePrint Arch.*, page 280, 2021. URL: https://eprint.iacr.org/2021/280. (Cited on page 33.)

[DFNS14] Ivan Damgård, Jakob Funder, Jesper Buus Nielsen, and Louis Salvail. Superposition attacks on cryptographic protocols. In Carles Padró, editor, *ICITS 13*, volume 8317 of *LNCS*, pages 142–161. Springer, Heidelberg, 2014. doi:10.1007/978-3-319-04268-8_9. (Cited on pages 189 and 193.)

[DGG16] Özgür Dagdelen, Sebastian Gajek, and Florian Göpfert. Learning with errors in the exponent. In Soonhak Kwon and Aaram Yun, editors, *ICISC 15*, volume 9558 of *LNCS*, pages 69–84. Springer, Heidelberg, November 2016. doi:10.1007/978-3-319-30840-1_5. (Cited on page 35.)

[DHP+18] Joan Daemen, Seth Hoffert, Michaël Peeters, Gilles Van Assche, and Ronny Van Keer. Xoodoo cookbook. Cryptology ePrint Archive, Report 2018/767, 2018. https://eprint.iacr.org/2018/767. (Cited on page 101.)

[Die82] D. Dieks. Communication by epr devices. *Physics Letters A*, 92(6):271–272, 1982. doi:https://doi.org/10.1016/0375-9601(82)90084-6. (Cited on pages 31, 152, and 217.)

[Dir39] P. A. M. Dirac. A new notation for quantum mechanics. *Mathematical Proceedings of the Cambridge Philosophical Society*, 35(3):416–418, 1939. doi:10.1017/S0305004100021162. (Cited on page 30.)

[DJS19] Jean Paul Degabriele, Christian Janson, and Patrick Struck. Sponges resist leakage: The case of authenticated encryption. In Steven D. Galbraith and Shiho Moriai, editors, *ASIACRYPT 2019, Part II*, volume 11922 of *LNCS*, pages 209–240. Springer, Heidelberg, December 2019. doi:10.1007/978-3-030-34621-8_8. (Cited on pages 4 and 43.)

[DKK07] Orr Dunkelman, Nathan Keller, and Jongsung Kim. Related-key rectangle attack on the full SHACAL-1. In Eli Biham and Amr M. Youssef, editors, *SAC 2006*, volume 4356 of *LNCS*, pages 28–44. Springer, Heidelberg, August 2007. doi:10.1007/978-3-540-74462-7_3. (Cited on pages 104 and 106.)

[DKR+20] Jan-Pieter D'Anvers, Angshuman Karmakar, Sujoy Sinha Roy, Frederik Vercauteren, Jose Maria Bermudo Mera, Michiel Van Beirendonck, and Andrea Basso. SABER. Technical report, National Institute of Standards and Technology, 2020. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions. (Cited on page 35.)

[DM19] Christoph Dobraunig and Bart Mennink. Leakage resilience of the duplex construction. In Steven D. Galbraith and Shiho Moriai, editors, *ASIACRYPT 2019, Part III*, volume 11923 of *LNCS*, pages 225–255. Springer, Heidelberg, December 2019. doi:10.1007/978-3-030-34618-8_8. (Cited on pages 44 and 45.)

[DM21] Christoph Dobraunig and Bart Mennink. Leakage resilient value comparison with application to message authentication. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part II*, volume 12697 of *LNCS*, pages 377–407. Springer, Heidelberg, October 2021. doi:10.1007/978-3-030-77886-6_13. (Cited on pages 45 and 101.)

[DMA17] Joan Daemen, Bart Mennink, and Gilles Van Assche. Full-state keyed duplex with built-in multi-user support. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT 2017, Part II*, volume 10625 of *LNCS*, pages 606–637. Springer, Heidelberg, December 2017. doi:10.1007/978-3-319-70697-9_21. (Cited on page 21.)

[DNR04] Cynthia Dwork, Moni Naor, and Omer Reingold. Immunizing encryption schemes from decryption errors. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 342–360. Springer, Heidelberg, May 2004. doi:10.1007/978-3-540-24676-3_21. (Cited on page 16.)

[DORS08] Yevgeniy Dodis, Rafail Ostrovsky, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM journal on computing*, 38(1):97–139, 2008. (Cited on page 27.)

[DP08] Stefan Dziembowski and Krzysztof Pietrzak. Leakage-resilient cryptography. In *49th FOCS*, pages 293–302. IEEE Computer Society Press, October 2008. doi:10.1109/FOCS.2008.56. (Cited on pages 21, 43, and 45.)

[DP10] Yevgeniy Dodis and Krzysztof Pietrzak. Leakage-resilient pseudorandom functions and side-channel attacks on Feistel networks. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 21–40. Springer, Heidelberg, August 2010. doi:10.1007/978-3-642-14623-7_2. (Cited on pages 44 and 45.)

[EEE20] Muhammed F. Esgin, Oguzhan Ersoy, and Zekeriya Erkin. Post-quantum adaptor signatures and payment channel networks. In Liqun Chen, Ninghui Li, Kaitai Liang, and Steve A. Schneider, editors, *ESORICS 2020, Part II*, volume 12309 of *LNCS*, pages 378–397. Springer, Heidelberg, September 2020. doi:10.1007/978-3-030-59013-0_19. (Cited on pages 153 and 171.)

[EKM+08] Thomas Eisenbarth, Timo Kasper, Amir Moradi, Christof Paar, Mahmoud Salmasizadeh, and Mohammad T. Manzuri Shalmani. On the power of power analysis in the real world: A complete break of the keeloqcode hopping scheme. In David Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 203–220. Springer, Heidelberg, August 2008. doi:10.1007/978-3-540-85174-5_12. (Cited on page 43.)

[ElG85] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31:469–472, 1985. (Cited on pages 128 and 151.)

[EM97] Shimon Even and Yishay Mansour. A construction of a cipher from a single pseudorandom permutation. *Journal of Cryptology*, 10(3):151–162, June 1997. doi:10.1007/s001459900025. (Cited on page 154.)

[ESZ21] Muhammed F. Esgin, Ron Steinfeld, and Raymond K. Zhao. Matrict+: More efficient post-quantum private blockchain payments. *IACR Cryptol. ePrint Arch.*, page 545, 2021. URL: https://eprint.iacr.org/2021/545. (Cited on page 171.)

[EZS+19] Muhammed F. Esgin, Raymond K. Zhao, Ron Steinfeld, Joseph K. Liu, and Dongxi Liu. MatRiCT: Efficient, scalable and post-quantum blockchain confidential transactions protocol. In Lorenzo Cavallaro, Johannes Kinder, Xi-

aoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 567–584. ACM Press, November 2019. doi:10.1145/3319535.3354200. (Cited on page 171.)

[FIP15] PUB FIPS. 202. *Federal Information Processing Standards Publication 202: SHA-3Standard: Permutation-Based Hash and Extendable-Output Functions*, 2015. (Cited on page 155.)

[FKM+16] Nils Fleischhacker, Johannes Krupp, Giulio Malavolta, Jonas Schneider, Dominique Schröder, and Mark Simkin. Efficient unlinkable sanitizable signatures from signatures with re-randomizable keys. In Chen-Mou Cheng, Kai-Min Chung, Giuseppe Persiano, and Bo-Yin Yang, editors, *PKC 2016, Part I*, volume 9614 of *LNCS*, pages 301–330. Springer, Heidelberg, March 2016. doi:10.1007/978-3-662-49384-7_12. (Cited on page 17.)

[FKOS22] Sebastian Faust, Juliane Krämer, Maximilian Orlt, and Patrick Struck. Related-key attack security of nonce-based authenticated encryption. *IACR Cryptol. ePrint Arch.*, 2022:xxx, 2022. URL: https://eprint.iacr.org/2022/xxx. (Cited on pages 5 and 103.)

[FM21] Marc Fischlin and Arno Mittelbach. An overview of the hybrid argument. *IACR Cryptol. ePrint Arch.*, 2021:88, 2021. URL: https://eprint.iacr.org/2021/088. (Cited on pages 50, 87, 97, 124, 140, 143, and 164.)

[FO99] Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In Michael J. Wiener, editor, *CRYPTO'99*, volume 1666 of *LNCS*, pages 537–554. Springer, Heidelberg, August 1999. doi:10.1007/3-540-48405-1_34. (Cited on page 186.)

[FO13] Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. *Journal of Cryptology*, 26(1):80–101, January 2013. doi:10.1007/s00145-011-9114-1. (Cited on page 186.)

[FPS12] Sebastian Faust, Krzysztof Pietrzak, and Joachim Schipper. Practical leakage-resilient symmetric cryptography. In Emmanuel Prouff and Patrick Schaumont, editors, *CHES 2012*, volume 7428 of *LNCS*, pages 213–232. Springer, Heidelberg, September 2012. doi:10.1007/978-3-642-33027-8_13. (Cited on pages 22, 44, and 45.)

[FS87] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 186–194. Springer, Heidelberg, August 1987. doi:10.1007/3-540-47721-7_12. (Cited on page 152.)

[Gag17] Tommaso Gagliardoni. *Quantum Security of Cryptographic Primitives*. PhD thesis, Darmstadt University of Technology, Germany, 2017. (Cited on pages 179, 189, 190, 193, and 215.)

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *41st ACM STOC*, pages 169–178. ACM Press, May / June 2009. doi:10.1145/1536414.1536440. (Cited on page 216.)

[GHHM21] Alex B. Grilo, Kathrin Hövelmanns, Andreas Hülsing, and Christian Majenz. Tight adaptive reprogramming in the QROM. In Mehdi Tibouchi and Huaxiong Wang, editors, *ASIACRYPT 2021*, volume 13090 of *Lecture Notes in Computer Science*, pages 637–667. Springer, 2021. doi:10.1007/978-3-030-92062-3_22. (Cited on page 155.)

[GHS16] Tommaso Gagliardoni, Andreas Hülsing, and Christian Schaffner. Semantic security and indistinguishability in the quantum world. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part III*, volume 9816 of *LNCS*, pages 60–89. Springer, Heidelberg, August 2016. doi:10.1007/978-3-662-53015-3_3. (Cited on pages 31, 33, 190, 191, 192, 193, 194, 195, 196, 197, 199, 200, 201, 202, 211, 213, and 220.)

[GKS21] Tommaso Gagliardoni, Juliane Krämer, and Patrick Struck. Quantum indistinguishability for public key encryption. In Jung Hee Cheon and Jean-Pierre Tillich, editors, *Post-Quantum Cryptography - 12th International Workshop, PQCrypto 2021, Daejeon, South Korea, July 20-22, 2021, Proceedings*, volume 12841 of *Lecture Notes in Computer Science*, pages 463–482. Springer, 2021. doi:10.1007/978-3-030-81293-5_24. (Cited on pages 6 and 189.)

[GKZ19] Alex B. Grilo, Iordanis Kerenidis, and Timo Zijlstra. Learning-with-errors problem is easy with quantum samples. *Phys. Rev. A*, 99:032314, Mar 2019. URL: https://link.aps.org/doi/10.1103/PhysRevA.99.032314, doi:10.1103/PhysRevA.99.032314. (Cited on page 217.)

[GLNP18] Shay Gueron, Yehuda Lindell, Ariel Nof, and Benny Pinkas. Fast garbling of circuits under standard assumptions. *Journal of Cryptology*, 31(3):798–844, July 2018. doi:10.1007/s00145-017-9271-y. (Cited on page 187.)

[GMO01] Karine Gandolfi, Christophe Mourtel, and Francis Olivier. Electromagnetic analysis: Concrete results. In Çetin Kaya Koç, David Naccache, and Christof Paar, editors, *CHES 2001*, volume 2162 of *LNCS*, pages 251–261. Springer, Heidelberg, May 2001. doi:10.1007/3-540-44709-1_21. (Cited on page 43.)

[GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, 1989. (Cited on page 152.)

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In Alfred Aho, editor, *19th ACM STOC*, pages 218–229. ACM Press, May 1987. doi:10.1145/28395.28420. (Cited on page 179.)

[Gol09] Oded Goldreich. *Foundations of cryptography: volume 2, basic applications*. Cambridge university press, 2009. (Cited on page 104.)

[Göp16] Florian Göpfert. *Securely instantiating cryptographic schemes based on the learning with errors assumption*. PhD thesis, Darmstadt University of Technology, Germany, 2016. (Cited on page 35.)

[GP99] Louis Goubin and Jacques Patarin. DES and differential power analysis (the "duplication" method). In Çetin Kaya Koç and Christof Paar, editors, *CHES'99*, volume 1717 of *LNCS*, pages 158–172. Springer, Heidelberg, August 1999. doi:10.1007/3-540-48059-5_15. (Cited on page 45.)

[GPPS19] Chun Guo, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. Authenticated encryption with nonce misuse and physical leakage: Definitions, separation results and first construction - (extended abstract). In Peter Schwabe and Nicolas Thériault, editors, *LATINCRYPT 2019*, volume 11774 of *LNCS*, pages 150–172. Springer, Heidelberg, 2019. doi:10.1007/978-3-030-30530-7_8. (Cited on pages 45 and 90.)

[GPPS20] Chun Guo, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. Towards low-energy leakage-resistant AE from the duplex sponge. *IACR Trans. Symm. Cryptol.*, 2020(1):6–42, 2020. doi:10.13154/tosc.v2020.i1.6-42. (Cited on pages 44 and 45.)

[GPT15] Peter Gaži, Krzysztof Pietrzak, and Stefano Tessaro. Tight bounds for keyed sponges and truncated CBC. Cryptology ePrint Archive, Report 2015/053, 2015. https://eprint.iacr.org/2015/053. (Cited on page 20.)

[GR05] Tal Garfinkel and Mendel Rosenblum. When virtual is harder than real: Security challenges in virtual machine based computing environments. In *Proceedings of HotOS'05: 10th Workshop on Hot Topics in Operating Systems, June 12-15, 2005, Santa Fe, New Mexico, USA*. USENIX Association, 2005. URL: http://www.usenix.org/events/hotos05/final_papers/full_papers/garfinkel/garfinkel.pdf. (Cited on page 106.)

[Gro96] Lov K. Grover. A fast quantum mechanical algorithm for database search. In *28th ACM STOC*, pages 212–219. ACM Press, May 1996. doi:10.1145/237814.237866. (Cited on page 151.)

[GSWY19] Chun Guo, François-Xavier Standaert, Weijia Wang, and Yu Yu. Efficient side-channel secure message authentication with better bounds. *IACR Trans. Symm. Cryptol.*, 2019(4):23–53, 2019. doi:10.13154/tosc.v2019.i4.23-53. (Cited on page 44.)

[GYZ17] Sumegha Garg, Henry Yuen, and Mark Zhandry. New security notions and feasibility results for authentication of quantum data. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part II*, volume 10402 of *LNCS*, pages 342–371. Springer, Heidelberg, August 2017. doi:10.1007/978-3-319-63715-0_12. (Cited on page 191.)

[GZB+19] Oscar Garcia-Morchon, Zhenfei Zhang, Sauvik Bhattacharya, Ronald Rietman, Ludo Tolhuizen, Jose-Luis Torre-Arce, Hayo Baan, Markku-Juhani O. Saarinen, Scott Fluhrer, Thijs Laarhoven, and Rachel Player. Round5. Technical report, National Institute of Standards and Technology, 2019. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions. (Cited on page 35.)

[Har11] David G. Harris. Critique of the related-key attack concept. *Des. Codes Cryptogr.*, 59(1-3):159–168, 2011. doi:10.1007/s10623-010-9451-3. (Cited on pages 27 and 105.)

[HHK17] Dennis Hofheinz, Kathrin Hövelmanns, and Eike Kiltz. A modular analysis of the Fujisaki-Okamoto transformation. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part I*, volume 10677 of *LNCS*, pages 341–371. Springer, Heidelberg, November 2017. doi:10.1007/978-3-319-70500-2_12. (Cited on page 186.)

[HKLN20] Eduard Hauck, Eike Kiltz, Julian Loss, and Ngoc Khanh Nguyen. Lattice-based blind signatures, revisited. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part II*, volume 12171 of *LNCS*, pages 500–529. Springer, Heidelberg, August 2020. doi:10.1007/978-3-030-56880-1_18. (Cited on page 153.)

[HKSU20] Kathrin Hövelmanns, Eike Kiltz, Sven Schäge, and Dominique Unruh. Generic authenticated key exchange in the quantum random oracle model. In Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas, editors, *PKC 2020, Part II*, volume 12111 of *LNCS*, pages 389–422. Springer, Heidelberg, May 2020. doi:10.1007/978-3-030-45388-6_14. (Cited on pages 33 and 186.)

[HLC+18] Zhengan Huang, Junzuo Lai, Wenbin Chen, Man Ho Au, Zhen Peng, and Jin Li. Hedged nonce-based public-key encryption: Adaptive security under randomness failures. In Michel Abdalla and Ricardo Dahab, editors, *PKC 2018, Part I*, volume 10769 of *LNCS*, pages 253–279. Springer, Heidelberg, March 2018. doi:10.1007/978-3-319-76578-5_9. (Cited on page 106.)

[HLL16] Shuai Han, Shengli Liu, and Lin Lyu. Efficient KDM-CCA secure public-key encryption for polynomial functions. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part II*, volume 10032 of *LNCS*, pages 307–338. Springer, Heidelberg, December 2016. doi:10.1007/978-3-662-53890-6_11. (Cited on pages 104 and 106.)

[HS18] Akinori Hosoyamada and Yu Sasaki. Quantum Demiric-Selçuk meet-in-the-middle attacks: Applications to 6-round generic Feistel constructions. In Dario Catalano and Roberto De Prisco, editors, *SCN 18*, volume 11035 of *LNCS*, pages 386–403. Springer, Heidelberg, September 2018. doi:10.1007/978-3-319-98113-0_21. (Cited on page 193.)

[HSS11] Sean Hallgren, Adam Smith, and Fang Song. Classical cryptographic protocols in a quantum world. In Phillip Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 411–428. Springer, Heidelberg, August 2011. doi:10.1007/978-3-642-22792-9_23. (Cited on page 155.)

[HXY19] Minki Hhan, Keita Xagawa, and Takashi Yamakawa. Quantum random oracle model with auxiliary input. In Steven D. Galbraith and Shiho Moriai, editors, *ASIACRYPT 2019, Part I*, volume 11921 of *LNCS*, pages 584–614. Springer, Heidelberg, December 2019. doi:10.1007/978-3-030-34578-5_21. (Cited on page 33.)

[IHM+19] Gembu Ito, Akinori Hosoyamada, Ryutaroh Matsumoto, Yu Sasaki, and Tetsu Iwata. Quantum chosen-ciphertext attacks against Feistel ciphers. In Mitsuru Matsui, editor, *CT-RSA 2019*, volume 11405 of *LNCS*, pages 391–411. Springer, Heidelberg, March 2019. doi:10.1007/978-3-030-12612-4_20. (Cited on page 193.)

[IIMP19] Akiko Inoue, Tetsu Iwata, Kazuhiko Minematsu, and Bertram Poettering. Cryptanalysis of OCB2: Attacks on authenticity and confidentiality. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part I*, volume 11692 of *LNCS*, pages 3–31. Springer, Heidelberg, August 2019. doi:10.1007/978-3-030-26948-7_1. (Cited on page 2.)

[IM18] Akiko Inoue and Kazuhiko Minematsu. Cryptanalysis of OCB2. Cryptology ePrint Archive, Report 2018/1040, 2018. https://eprint.iacr.org/2018/1040. (Cited on page 2.)

[Iso11] Takanori Isobe. A single-key attack on the full GOST block cipher. In Antoine Joux, editor, *FSE 2011*, volume 6733 of *LNCS*, pages 290–305. Springer, Heidelberg, February 2011. doi:10.1007/978-3-642-21702-9_17. (Cited on pages 104 and 106.)

[Iwa18] Tetsu Iwata. Plaintext recovery attack of OCB2. Cryptology ePrint Archive, Report 2018/1090, 2018. https://eprint.iacr.org/2018/1090. (Cited on page 2.)

[JDF11] David Jao and Luca De Feo. Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. In *International Workshop on Post-Quantum Cryptography*, pages 19–34. Springer, 2011. (Cited on page 152.)

[JLM14] Philipp Jovanovic, Atul Luykx, and Bart Mennink. Beyond $2^{c/2}$ security in sponge-based authenticated encryption modes. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Part I*, volume 8873 of *LNCS*, pages 85–104. Springer, Heidelberg, December 2014. doi:10.1007/978-3-662-45611-8_5. (Cited on page 20.)

[JNPS16] Jérémy Jean, Ivica Nikolić, Thomas Peyrin, and Yannick Seurin. Deoxys v1.41. *Submission to the CAESAR Competition*, 2016. (Cited on page 147.)

[JS22] Christian Janson and Patrick Struck. Sponge-based authenticated encryption: Security against quantum attackers. *IACR Cryptol. ePrint Arch.*, 2022:xxx, 2022. URL: https://eprint.iacr.org/2022/xxx. (Cited on pages 5 and 151.)

[JST21] Joseph Jaeger, Fang Song, and Stefano Tessaro. Quantum key-length extension. In Kobbi Nissim and Brent Waters, editors, *TCC 2021*, volume 13042 of *Lecture Notes in Computer Science*, pages 209–239. Springer, 2021. URL: https://doi.org/10.1007/978-3-030-90459-3_8. (Cited on page 154.)

[JZC+18] Haodong Jiang, Zhenfeng Zhang, Long Chen, Hong Wang, and Zhi Ma. IND-CCA-secure key encapsulation mechanism in the quantum random oracle model, revisited. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part III*, volume 10993 of *LNCS*, pages 96–125. Springer, Heidelberg, August 2018. doi:10.1007/978-3-319-96878-0_4. (Cited on page 186.)

[JZM19a] Haodong Jiang, Zhenfeng Zhang, and Zhi Ma. Key encapsulation mechanism with explicit rejection in the quantum random oracle model. In Dongdai Lin and Kazue Sako, editors, *PKC 2019, Part II*, volume 11443 of *LNCS*, pages 618–645. Springer, Heidelberg, April 2019. doi:10.1007/978-3-030-17259-6_21. (Cited on page 33.)

[JZM19b] Haodong Jiang, Zhenfeng Zhang, and Zhi Ma. Tighter security proofs for generic key encapsulation mechanism in the quantum random oracle model. In Jintai Ding and Rainer Steinwandt, editors, *Post-Quantum Cryptography - 10th International Conference, PQCrypto 2019*, pages 227–248. Springer, Heidelberg, 2019. doi:10.1007/978-3-030-25510-7_13. (Cited on page 33.)

[KD13] Cameron F. Kerry and Charles Romine Director. Fips pub 186-4 federal information processing standards publication digital signature standard (dss). 2013. (Cited on page 151.)

[KHK11] Bonwook Koo, Deukjo Hong, and Daesung Kwon. Related-key attack on the full HIGHT. In Kyung Hyune Rhee and DaeHun Nyang, editors, *ICISC 10*, volume 6829 of *LNCS*, pages 49–67. Springer, Heidelberg, December 2011. (Cited on pages 104 and 106.)

[KJJ99] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In Michael J. Wiener, editor, *CRYPTO'99*, volume 1666 of *LNCS*, pages 388–397. Springer, Heidelberg, August 1999. doi:10.1007/3-540-48405-1_25. (Cited on page 43.)

[KK03] Lars R. Knudsen and Tadayoshi Kohno. Analysis of RMAC. In Thomas Johansson, editor, *FSE 2003*, volume 2887 of *LNCS*, pages 182–191. Springer, Heidelberg, February 2003. doi:10.1007/978-3-540-39887-5_14. (Cited on page 104.)

[KK08] Seny Kamara and Jonathan Katz. How to encrypt with a malicious random number generator. In Kaisa Nyberg, editor, *FSE 2008*, volume 5086 of *LNCS*, pages 303–315. Springer, Heidelberg, February 2008. doi:10.1007/978-3-540-71039-4_19. (Cited on page 106.)

[KKPP20] Shuichi Katsumata, Kris Kwiatkowski, Federico Pintore, and Thomas Prest. Scalable ciphertext compression techniques for post-quantum KEMs and their applications. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020, Part I*, volume 12491 of *LNCS*, pages 289–320. Springer, Heidelberg, December 2020. doi:10.1007/978-3-030-64837-4_10. (Cited on page 33.)

[KKVB02] Elham Kashefi, Adrian Kent, Vlatko Vedral, and Konrad Banaszek. Comparison of quantum oracles. *Phys. Rev. A*, 65:050304, May 2002. URL: https://link.aps.org/doi/10.1103/PhysRevA.65.050304, doi:10.1103/PhysRevA.65.050304. (Cited on pages 31 and 191.)

[KL20] Jonathan Katz and Yehuda Lindell. *Introduction to modern cryptography*. CRC press, 2020. (Cited on pages 2 and 104.)

[KLLN16a] Marc Kaplan, Gaëtan Leurent, Anthony Leverrier, and María Naya-Plasencia. Breaking symmetric cryptosystems using quantum period finding. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part II*, volume 9815 of *LNCS*, pages 207–237. Springer, Heidelberg, August 2016. doi:10.1007/978-3-662-53008-5_8. (Cited on pages 190 and 193.)

[KLLN16b] Marc Kaplan, Gaëtan Leurent, Anthony Leverrier, and María Naya-Plasencia. Quantum differential and linear cryptanalysis. *IACR Trans. Symm. Cryptol.*, 2016(1):71–94, 2016. https://tosc.iacr.org/index.php/ToSC/article/view/536. doi:10.13154/tosc.v2016.i1.71-94. (Cited on page 189.)

[KLS18] Eike Kiltz, Vadim Lyubashevsky, and Christian Schaffner. A concrete treatment of Fiat-Shamir signatures in the quantum random-oracle model. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part III*, volume 10822 of *LNCS*, pages 552–586. Springer, Heidelberg, April / May 2018. doi:10.1007/978-3-319-78372-7_18. (Cited on page 155.)

[KM10] Hidenori Kuwakado and Masakatu Morii. Quantum distinguisher between the 3-round feistel cipher and the random permutation. In *IEEE International Symposium on Information Theory, ISIT 2010, June 13-18, 2010, Austin, Texas, USA, Proceedings*, pages 2682–2685, 2010. URL: https://doi.org/10.1109/ISIT.2010.5513654. (Cited on page 193.)

[KM12] Hidenori Kuwakado and Masakatu Morii. Security on the quantum-type even-mansour cipher. In *Proceedings of the International Symposium on Information Theory and its Applications, ISITA 2012, Honolulu, HI, USA, October*

28-31, 2012, pages 312–316, 2012. URL: http://ieeexplore.ieee.org/document/6400943/. (Cited on pages 190 and 193.)

[KMR14] Vladimir Kolesnikov, Payman Mohassel, and Mike Rosulek. FleXOR: Flexible garbling for XOR gates that beats free-XOR. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part II*, volume 8617 of *LNCS*, pages 440–457. Springer, Heidelberg, August 2014. doi:10.1007/978-3-662-44381-1_25. (Cited on page 187.)

[Knu93] Lars R. Knudsen. Cryptanalysis of LOKI91. In Jennifer Seberry and Yuliang Zheng, editors, *AUSCRYPT'92*, volume 718 of *LNCS*, pages 196–208. Springer, Heidelberg, December 1993. doi:10.1007/3-540-57220-1_62. (Cited on pages 104 and 105.)

[Koc96] Paul C. Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In Neal Koblitz, editor, *CRYPTO'96*, volume 1109 of *LNCS*, pages 104–113. Springer, Heidelberg, August 1996. doi:10.1007/3-540-68697-5_9. (Cited on pages 3 and 43.)

[KPG99] Aviad Kipnis, Jacques Patarin, and Louis Goubin. Unbalanced oil and vinegar signature schemes. In Jacques Stern, editor, *EUROCRYPT'99*, volume 1592 of *LNCS*, pages 206–222. Springer, Heidelberg, May 1999. doi:10.1007/3-540-48910-X_15. (Cited on page 152.)

[KR01] Joe Kilian and Phillip Rogaway. How to protect DES against exhaustive key search (an analysis of DESX). *Journal of Cryptology*, 14(1):17–35, January 2001. doi:10.1007/s001450010015. (Cited on page 154.)

[Krä15] Juliane Krämer. *Why cryptography should not rely on physical attack complexity*. PhD thesis, Berlin Institute of Technology, 2015. (Cited on page 220.)

[KS08] Vladimir Kolesnikov and Thomas Schneider. Improved garbled circuit: Free XOR gates and applications. In Luca Aceto, Ivan Damgård, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfsdóttir, and Igor Walukiewicz, editors, *ICALP 2008, Part II*, volume 5126 of *LNCS*, pages 486–498. Springer, Heidelberg, July 2008. doi:10.1007/978-3-540-70583-3_40. (Cited on page 187.)

[KS20a] Juliane Krämer and Patrick Struck. Encryption schemes using random oracles: From classical to post-quantum security. In Jintai Ding and Jean-Pierre Tillich, editors, *Post-Quantum Cryptography - 11th International Conference, PQCrypto 2020*, pages 539–558. Springer, Heidelberg, 2020. doi:10.1007/978-3-030-44223-1_29. (Cited on pages 5 and 151.)

[KS20b] Juliane Krämer and Patrick Struck. Leakage-resilient authenticated encryption from leakage-resilient pseudorandom functions. In Guido Marco Bertoni and Francesco Regazzoni, editors, *COSADE 2020*, volume 12244 of *LNCS*, pages 315–337. Springer, Heidelberg, April 2020. doi:10.1007/978-3-030-68773-1_15. (Cited on pages 4 and 43.)

[KS20c] Juliane Krämer and Patrick Struck. Security of public key encryption against resetting attacks. In Karthikeyan Bhargavan, Elisabeth Oswald, and Manoj Prabhakaran, editors, *INDOCRYPT 2020*, volume 12578 of *LNCS*, pages 508–528. Springer, Heidelberg, December 2020. doi:10.1007/978-3-030-65277-7_23. (Cited on pages 5 and 103.)

[KSS+20] Veronika Kuchta, Amin Sakzad, Damien Stehlé, Ron Steinfeld, and Shifeng Sun. Measure-rewind-measure: Tighter quantum random oracle model proofs for one-way to hiding and CCA security. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part III*, volume 12107 of *LNCS*, pages 703–728. Springer, Heidelberg, May 2020. doi:10.1007/978-3-030-45727-3_24. (Cited on pages 32, 155, and 187.)

[Lam79] Leslie Lamport. Constructing digital signatures from a one-way function. Technical Report SRI-CSL-98, SRI International Computer Science Laboratory, October 1979. (Cited on page 152.)

[LAZ19] Xingye Lu, Man Ho Au, and Zhenfei Zhang. Raptor: A practical lattice-based (linkable) ring signature. In Robert H. Deng, Valérie Gauthier-Umaña, Martín Ochoa, and Moti Yung, editors, *ACNS 19*, volume 11464 of *LNCS*, pages 110–130. Springer, Heidelberg, June 2019. doi:10.1007/978-3-030-21568-2_6. (Cited on page 153.)

[Lin17] Yehuda Lindell. How to simulate it - A tutorial on the simulation proof technique. In Yehuda Lindell, editor, *Tutorials on the Foundations of Cryptography*, pages 277–346. Springer International Publishing, 2017. doi:10.1007/978-3-319-57048-8_6. (Cited on pages 2 and 181.)

[LLJ15] Xianhui Lu, Bao Li, and Dingding Jia. KDM-CCA security from RKA secure authenticated encryption. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 559–583. Springer, Heidelberg, April 2015. doi:10.1007/978-3-662-46800-5_22. (Cited on pages 104, 106, and 107.)

[LLJ+19] Xianhui Lu, Yamin Liu, Dingding Jia, Haiyang Xue, Jingnan He, Zhenfei Zhang, Zhe Liu, Hao Yang, Bao Li, and Kunpeng Wang. LAC. Technical report, National Institute of Standards and Technology, 2019. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions. (Cited on page 35.)

[LM17] Gregor Leander and Alexander May. Grover meets simon - quantumly attacking the FX-construction. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT 2017, Part II*, volume 10625 of *LNCS*, pages 161–178. Springer, Heidelberg, December 2017. doi:10.1007/978-3-319-70697-9_6. (Cited on page 193.)

[LMO+14] Jake Longo, Daniel P. Martin, Elisabeth Oswald, Daniel Page, Martijn Stam, and Michael Tunstall. Simulatable leakage: Analysis, pitfalls, and new constructions. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Part I*, volume 8873 of *LNCS*, pages 223–242. Springer, Heidelberg, December 2014. doi:10.1007/978-3-662-45611-8_12. (Cited on page 45.)

[LN17] Vadim Lyubashevsky and Gregory Neven. One-shot verifiable encryption from lattices. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part I*, volume 10210 of *LNCS*, pages 293–323. Springer, Heidelberg, April / May 2017. doi:10.1007/978-3-319-56620-7_11. (Cited on page 153.)

[LP09] Yehuda Lindell and Benny Pinkas. A proof of security of Yao's protocol for two-party computation. *Journal of Cryptology*, 22(2):161–188, April 2009. doi:10.1007/s00145-008-9036-8. (Cited on pages 154, 179, 180, 181, 182, 183, and 186.)

[LP11] Richard Lindner and Chris Peikert. Better key sizes (and attacks) for LWE-based encryption. In Aggelos Kiayias, editor, *CT-RSA 2011*, volume 6558 of *LNCS*, pages 319–339. Springer, Heidelberg, February 2011. doi:10.1007/978-3-642-19074-2_21. (Cited on page 35.)

[LS15] Adeline Langlois and Damien Stehlé. Worst-case to average-case reductions for module lattices. *Designs, Codes and Cryptography*, 75(3):565–599, 2015. (Cited on page 35.)

[LW21] Xu Liu and Mingqiang Wang. QCCA-secure generic key encapsulation mechanism with tighter security in the quantum random oracle model. In Juan Garay, editor, *PKC 2021, Part I*, volume 12710 of *LNCS*, pages 3–26. Springer, Heidelberg, May 2021. doi:10.1007/978-3-030-75245-3_1. (Cited on page 33.)

[Lyu09] Vadim Lyubashevsky. Fiat-Shamir with aborts: Applications to lattice and factoring-based signatures. In Mitsuru Matsui, editor, *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 598–616. Springer, Heidelberg, December 2009. doi:10.1007/978-3-642-10366-7_35. (Cited on page 152.)

[Lyu12] Vadim Lyubashevsky. Lattice signatures without trapdoors. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 738–755. Springer, Heidelberg, April 2012. doi:10.1007/978-3-642-29011-4_43. (Cited on page 152.)

[LZ19] Qipeng Liu and Mark Zhandry. Revisiting post-quantum Fiat-Shamir. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part II*, volume 11693 of *LNCS*, pages 326–355. Springer, Heidelberg, August 2019. doi:10.1007/978-3-030-26951-7_12. (Cited on page 155.)

[Mar20] Varun Maram. On the security of NTS-KEM in the quantum random oracle model. In Marco Baldi, Edoardo Persichetti, and Paolo Santini, editors, *Code-Based Cryptography - 8th International Workshop, CBCrypto 2020, Zagreb, Croatia, May 9-10, 2020, Revised Selected Papers*, volume 12087 of *Lecture Notes in Computer Science*, pages 1–19. Springer, 2020. doi:10.1007/978-3-030-54074-6_1. (Cited on page 33.)

[McE78] Robert J. McEliece. A public-key cryptosystem based on algebraic coding theory. The deep space network progress report 42-44, Jet Propulsion Laboratory, California Institute of Technology, January/February 1978. https://ipnpr.jpl.nasa.gov/progress_report2/42-44/44N.PDF. (Cited on page 152.)

[Mer90] Ralph C. Merkle. A certified digital signature. In Gilles Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 218–238. Springer, Heidelberg, August 1990. doi:10.1007/0-387-34805-0_21. (Cited on page 152.)

[MF21] Arno Mittelbach and Marc Fischlin. *The Theory of Hash Functions and Random Oracles - An Approach to Modern Cryptography*. Information Security and Cryptography. Springer, 2021. doi:10.1007/978-3-030-63287-8. (Cited on pages 50, 87, 97, 104, 140, and 143.)

[MI88] Tsutomu Matsumoto and Hideki Imai. Public quadratic polynominal-tuples for efficient signature-verification and message-encryption. In C. G. Günther, editor, *EUROCRYPT'88*, volume 330 of *LNCS*, pages 419–453. Springer, Heidelberg, May 1988. doi:10.1007/3-540-45961-8_39. (Cited on page 152.)

[MR04] Silvio Micali and Leonid Reyzin. Physically observable cryptography (extended abstract). In Moni Naor, editor, *TCC 2004*, volume 2951 of *LNCS*, pages 278–296. Springer, Heidelberg, February 2004. doi:10.1007/978-3-540-24638-1_16. (Cited on page 21.)

[MRH04] Ueli M. Maurer, Renato Renner, and Clemens Holenstein. Indifferentiability, impossibility results on reductions, and applications to the random oracle methodology. In Moni Naor, editor, *TCC 2004*, volume 2951 of *LNCS*, pages 21–39. Springer, Heidelberg, February 2004. doi:10.1007/978-3-540-24638-1_2. (Cited on page 101.)

[MRV15] Bart Mennink, Reza Reyhanitabar, and Damian Vizár. Security of full-state keyed sponge and duplex: Applications to authenticated encryption. In Tetsu Iwata and Jung Hee Cheon, editors, *ASIACRYPT 2015, Part II*, volume 9453 of *LNCS*, pages 465–489. Springer, Heidelberg, November / December 2015. doi:10.1007/978-3-662-48800-3_19. (Cited on page 21.)

[MS16] Shahram Mossayebi and Rüdiger Schack. Concrete security against adversaries with quantum superposition access to encryption and decryption oracles. *CoRR*, abs/1609.03780, 2016. URL: http://arxiv.org/abs/1609.03780, arXiv:1609.03780. (Cited on pages 191, 193, 195, 213, and 220.)

[MS18] Takahiro Matsuda and Jacob C. N. Schuldt. Related randomness security for public key encryption, revisited. In Michel Abdalla and Ricardo Dahab, editors, *PKC 2018, Part I*, volume 10769 of *LNCS*, pages 280–311. Springer, Heidelberg, March 2018. doi:10.1007/978-3-319-76578-5_10. (Cited on pages 106 and 147.)

[MSGR10] Marcel Medwed, François-Xavier Standaert, Johann Großschädl, and Francesco Regazzoni. Fresh re-keying: Security against side-channel and fault attacks for low-cost devices. In Daniel J. Bernstein and Tanja Lange, editors, *AFRICACRYPT 10*, volume 6055 of *LNCS*, pages 279–296. Springer, Heidelberg, May 2010. (Cited on page 44.)

[MSJ12] Marcel Medwed, François-Xavier Standaert, and Antoine Joux. Towards super-exponential side-channel security with efficient leakage-resilient PRFs. In Emmanuel Prouff and Patrick Schaumont, editors, *CHES 2012*, volume 7428 of *LNCS*, pages 193–212. Springer, Heidelberg, September 2012. doi:10.1007/978-3-642-33027-8_12. (Cited on page 44.)

[MSNF16] Marcel Medwed, François-Xavier Standaert, Ventzislav Nikov, and Martin Feldhofer. Unknown-input attacks in the parallel setting: Improving the security of the CHES 2012 leakage-resilient PRF. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part I*, volume 10031 of *LNCS*, pages 602–623. Springer, Heidelberg, December 2016. doi:10.1007/978-3-662-53887-6_22. (Cited on page 44.)

[MSS21] Payman Mohassel, Seyed Saeed Sadeghian, and Nigel P. Smart. Actively secure private function evaluation. In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology - ASIACRYPT 2021 - 27th International Conference on the Theory and Application of Cryptology and Information Security, Virtual, December 6-10, 2021*. Springer, 2021. (Cited on page 33.)

[NAB+19] Michael Naehrig, Erdem Alkim, Joppe Bos, Léo Ducas, Karen Easterbrook, Brian LaMacchia, Patrick Longa, Ilya Mironov, Valeria Nikolaenko, Christopher Peikert, Ananth Raghunathan, and Douglas Stebila. FrodoKEM. Technical report, National Institute of Standards and Technology, 2019. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions. (Cited on page 35.)

[NC16] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information (10th Anniversary edition)*. Cambridge University Press, 2016. (Cited on pages 31, 191, and 195.)

[Nie86] H. Niederreiter. Knapsack-type cryptosystems and algebraic coding theory. *Problems of Control and Information Theory*, 15(2):159–166, 1986. (Cited on page 152.)

[NIST15] National Institute of Standards and Technology. Lightweight cryptography standardization process, 2015. (Cited on pages 44, 104, and 154.)

[NIST17] National Institute of Standards and Technology. Post-quantum cryptography standardization process, 2017. (Cited on pages 132, 152, 187, and 211.)

[NPS99] Moni Naor, Benny Pinkas, and Reuban Sumner. Privacy preserving auctions and mechanism design. In *Proceedings of the 1st ACM Conference on Electronic Commerce*, EC '99, page 129–139, New York, NY, USA, 1999. Association for Computing Machinery. doi:10.1145/336992.337028. (Cited on page 187.)

[NR98] Moni Naor and Omer Reingold. From unpredictability to indistinguishability: A simple construction of pseudo-random functions from MACs (extended abstract). In Hugo Krawczyk, editor, *CRYPTO'98*, volume 1462 of *LNCS*, pages 267–282. Springer, Heidelberg, August 1998. doi:10.1007/BFb0055734. (Cited on pages 46 and 89.)

[NRS14] Chanathip Namprempre, Phillip Rogaway, and Thomas Shrimpton. Reconsidering generic composition. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 257–274. Springer, Heidelberg, May 2014. doi:10.1007/978-3-642-55220-5_15. (Cited on pages 13, 14, 44, 105, 108, 109, and 223.)

[NY90] Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *22nd ACM STOC*, pages 427–437. ACM Press, May 1990. doi:10.1145/100216.100273. (Cited on page 216.)

[PAA+19] Thomas Poppelmann, Erdem Alkim, Roberto Avanzi, Joppe Bos, Léo Ducas, Antonio de la Piedra, Peter Schwabe, Douglas Stebila, Martin R. Albrecht, Emmanuela Orsini, Valery Osheter, Kenneth G. Paterson, Guy Peer, and Nigel P. Smart. NewHope. Technical report, National Institute of Standards and Technology, 2019. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions. (Cited on page 35.)

[Pat95] Jacques Patarin. Cryptanalysis of the Matsumoto and Imai public key scheme of eurocrypt'88. In Don Coppersmith, editor, *CRYPTO'95*, volume 963 of *LNCS*, pages 248–261. Springer, Heidelberg, August 1995. doi:10.1007/3-540-44750-4_20. (Cited on page 152.)

[Pat96] Jacques Patarin. Hidden fields equations (HFE) and isomorphisms of polynomials (IP): Two new families of asymmetric algorithms. In Ueli M. Maurer, editor, *EUROCRYPT'96*, volume 1070 of *LNCS*, pages 33–48. Springer, Heidelberg, May 1996. doi:10.1007/3-540-68339-9_4. (Cited on page 152.)

[Pie09] Krzysztof Pietrzak. A leakage-resilient mode of operation. In Antoine Joux, editor, *EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 462–482. Springer, Heidelberg, April 2009. doi:10.1007/978-3-642-01001-9_27. (Cited on page 44.)

[Poe18] Bertram Poettering. Breaking the confidentiality of OCB2. Cryptology ePrint Archive, Report 2018/1087, 2018. https://eprint.iacr.org/2018/1087. (Cited on page 2.)

[PS00] David Pointcheval and Jacques Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3):361–396, June 2000. doi:10.1007/s001450010003. (Cited on page 152.)

[PSP+08] Christophe Petit, François-Xavier Standaert, Olivier Pereira, Tal Malkin, and Moti Yung. A block cipher based pseudo random number generator secure against side-channel key recovery. In Masayuki Abe and Virgil Gligor, editors, *ASIACCS 08*, pages 56–65. ACM Press, March 2008. (Cited on page 44.)

[PSS14] Kenneth G. Paterson, Jacob C. N. Schuldt, and Dale L. Sibborn. Related randomness attacks for public key encryption. In Hugo Krawczyk, editor, *PKC 2014*, volume 8383 of *LNCS*, pages 465–482. Springer, Heidelberg, March 2014. doi:10.1007/978-3-642-54631-0_27. (Cited on pages 29, 104, 105, 106, 136, and 147.)

[PSSW09] Benny Pinkas, Thomas Schneider, Nigel P. Smart, and Stephen C. Williams. Secure two-party computation is practical. In Mitsuru Matsui, editor, *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 250–267. Springer, Heidelberg, December 2009. doi:10.1007/978-3-642-10366-7_15. (Cited on page 187.)

[PSV15] Olivier Pereira, François-Xavier Standaert, and Srinivas Vivek. Leakage-resilient authentication and encryption from symmetric cryptographic primitives. In Indrajit Ray, Ninghui Li, and Christopher Kruegel, editors, *ACM CCS 2015*, pages 96–108. ACM Press, October 2015. doi:10.1145/2810103.2813626. (Cited on page 45.)

[QS01] Jean-Jacques Quisquater and David Samyde. Electromagnetic analysis (ema): Measures and counter-measures for smart cards. In *International Conference on Research in Smart Cards*, pages 200–210. Springer, 2001. (Cited on page 43.)

[Rab79] Michael O. Rabin. Digital signatures and public key functions as intractable as factorization. Technical Report MIT/LCS/TR-212, Massachusetts Institute of Technology, January 1979. (Cited on page 3.)

[RBBK01] Phillip Rogaway, Mihir Bellare, John Black, and Ted Krovetz. OCB: A block-cipher mode of operation for efficient authenticated encryption. In Michael K. Reiter and Pierangela Samarati, editors, *ACM CCS 2001*, pages 196–205. ACM Press, November 2001. doi:10.1145/501983.502011. (Cited on page 10.)

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *37th ACM STOC*, pages 84–93. ACM Press, May 2005. doi:10.1145/1060590.1060603. (Cited on pages 35, 130, 131, 152, 192, and 203.)

[Reg09] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM*, 56(6):34:1–34:40, 2009. URL: http://doi.acm.org/10.1145/1568318.1568324, doi:10.1145/1568318.1568324. (Cited on pages 35, 130, 131, 152, 192, and 203.)

[Res18] Eric Rescorla. The Transport Layer Security (TLS) Protocol Version 1.3. RFC 8446, August 2018. URL: https://rfc-editor.org/rfc/rfc8446.txt, doi:10.17487/RFC8446. (Cited on page 104.)

[Rog02] Phillip Rogaway. Authenticated-encryption with associated-data. In Vijayalakshmi Atluri, editor, *ACM CCS 2002*, pages 98–107. ACM Press, November 2002. doi:10.1145/586110.586125. (Cited on pages 10, 11, and 104.)

[Rog04a] Phillip Rogaway. Efficient instantiations of tweakable blockciphers and refinements to modes OCB and PMAC. In Pil Joong Lee, editor, *ASIACRYPT 2004*, volume 3329 of *LNCS*, pages 16–31. Springer, Heidelberg, December 2004. doi:10.1007/978-3-540-30539-2_2. (Cited on page 2.)

[Rog04b] Phillip Rogaway. Nonce-based symmetric encryption. In Bimal K. Roy and Willi Meier, editors, *FSE 2004*, volume 3017 of *LNCS*, pages 348–359. Springer, Heidelberg, February 2004. doi:10.1007/978-3-540-25937-4_22. (Cited on pages 10 and 107.)

[RR01] Josyula R. Rao and Pankaj Rohatgi. EMpowering side-channel attacks. Cryptology ePrint Archive, Report 2001/037, 2001. https://eprint.iacr.org/2001/037. (Cited on page 43.)

[RS92] Charles Rackoff and Daniel R. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In Joan Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 433–444. Springer, Heidelberg, August 1992. doi:10.1007/3-540-46766-1_35. (Cited on page 217.)

[RS06a] Phillip Rogaway and Thomas Shrimpton. A provable-security treatment of the key-wrap problem. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 373–390. Springer, Heidelberg, May / June 2006. doi:10.1007/11761679_23. (Cited on page 107.)

[RS06b] Alexander Rostovtsev and Anton Stolbunov. Public-Key Cryptosystem Based On Isogenies. Cryptology ePrint Archive, Report 2006/145, 2006. https://eprint.iacr.org/2006/145. (Cited on page 152.)

[RS15] Martin Rötteler and Rainer Steinwandt. A note on quantum related-key attacks. *Inf. Process. Lett.*, 115(1):40–44, 2015. URL: https://doi.org/10.1016/j.ipl.2014.08.009. (Cited on page 193.)

[RSA78] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the Association for Computing Machinery*, 21(2):120–126, 1978. (Cited on pages 151 and 152.)

[RSSS17] Miruna Rosca, Amin Sakzad, Damien Stehlé, and Ron Steinfeld. Middle-product learning with errors. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part III*, volume 10403 of *LNCS*, pages 283–297. Springer, Heidelberg, August 2017. doi:10.1007/978-3-319-63697-9_10. (Cited on page 35.)

[RY10] Thomas Ristenpart and Scott Yilek. When good randomness goes bad: Virtual machine reset vulnerabilities and hedging deployed cryptography. In *NDSS 2010*. The Internet Society, February / March 2010. (Cited on pages 3 and 106.)

[SAB+20] Peter Schwabe, Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Gregor Seiler, and Damien Stehlé. CRYSTALS-KYBER. Technical report, National Institute of Standards and Technology, 2020. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions. (Cited on page 35.)

[SAL+17] Nigel P. Smart, Martin R. Albrecht, Yehuda Lindell, Emmanuela Orsini, Valery Osheter, Kenny Paterson, and Guy Peer. LIMA. Technical report, National Institute of Standards and Technology, 2017. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions. (Cited on page 35.)

[Sho94] Peter W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *35th FOCS*, pages 124–134. IEEE Computer Society Press, November 1994. doi:10.1109/SFCS.1994.365700. (Cited on pages 3 and 151.)

[Sho04] Victor Shoup. Sequences of games: a tool for taming complexity in security proofs. Cryptology ePrint Archive, Report 2004/332, 2004. https://eprint.iacr.org/2004/332. (Cited on page 8.)

[Shr04] Tom Shrimpton. A characterization of authenticated-encryption as a form of chosen-ciphertext security. Cryptology ePrint Archive, Report 2004/272, 2004. https://eprint.iacr.org/2004/272. (Cited on page 13.)

[Sib15] Dale Sibborn. *Analysis of Public-Key Encryption Schemes in Extended Attack Models*. PhD thesis, Royal Holloway, University of London, 2015. (Cited on page 136.)

[Son14] Fang Song. A note on quantum security for post-quantum cryptography. In Michele Mosca, editor, *Post-Quantum Cryptography - 6th International Workshop, PQCrypto 2014*, pages 246–265. Springer, Heidelberg, October 2014. doi:10.1007/978-3-319-11659-4_15. (Cited on pages 155 and 159.)

[SP06] Kai Schramm and Christof Paar. Higher order masking of the AES. In David Pointcheval, editor, *CT-RSA 2006*, volume 3860 of *LNCS*, pages 208–225. Springer, Heidelberg, February 2006. doi:10.1007/11605805_14. (Cited on page 45.)

[SPY+10] François-Xavier Standaert, Olivier Pereira, Yu Yu, Jean-Jacques Quisquater, Moti Yung, and Elisabeth Oswald. Leakage resilient cryptography in practice. In *Towards Hardware-Intrinsic Security - Foundations and Practice*, pages 99–134. Springer, 2010. doi:10.1007/978-3-642-14452-3_5. (Cited on pages 22, 45, and 220.)

[SPY13] François-Xavier Standaert, Olivier Pereira, and Yu Yu. Leakage-resilient symmetric cryptography under empirically verifiable assumptions. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 335–352. Springer, Heidelberg, August 2013. doi:10.1007/978-3-642-40041-4_19. (Cited on pages 44 and 45.)

[SRP18] Alan Szepieniec, Reza Reyhanitabar, and Bart Preneel. Key encapsulation from noisy key agreement in the quantum random oracle model. Cryptology ePrint Archive, Report 2018/884, 2018. https://eprint.iacr.org/2018/884. (Cited on page 33.)

[SSZ17] Ron Steinfeld, Amin Sakzad, and Raymond K. Zhao. Titanium. Technical report, National Institute of Standards and Technology, 2017. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions. (Cited on page 153.)

[SXY18] Tsunekazu Saito, Keita Xagawa, and Takashi Yamakawa. Tightly-secure key-encapsulation mechanism in the quantum random oracle model. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part III*, volume 10822 of *LNCS*, pages 520–551. Springer, Heidelberg, April / May 2018. doi:10.1007/978-3-319-78372-7_17. (Cited on page 186.)

[SY17] Fang Song and Aaram Yun. Quantum security of NMAC and related constructions - PRF domain extension against quantum attacks. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part II*, volume 10402 of *LNCS*, pages 283–309. Springer, Heidelberg, August 2017. doi:10.1007/978-3-319-63715-0_10. (Cited on page 33.)

[TKS+19] Wilson Abel Alberto Torres, Veronika Kuchta, Ron Steinfeld, Amin Sakzad, Joseph K. Liu, and Jacob Cheng. Lattice RingCT V2.0 with multiple input and multiple output wallets. In Julian Jang-Jaccard and Fuchun Guo, editors, *ACISP 19*, volume 11547 of *LNCS*, pages 156–175. Springer, Heidelberg, July 2019. doi:10.1007/978-3-030-21548-4_9. (Cited on page 153.)

[TMM21] Erkan Tairi, Pedro Moreno-Sanchez, and Matteo Maffei. Post-quantum adaptor signature for privacy-preserving off-chain payments. In Nikita Borisov and Claudia Diaz, editors, *Financial Cryptography and Data Security - 25th International Conference, FC 2021, Virtual Event, March 1-5, 2021, Revised Selected Papers, Part II*, volume 12675 of *Lecture Notes in Computer Science*, pages 131–150. Springer, 2021. doi:10.1007/978-3-662-64331-0_7. (Cited on page 171.)

[TSS+18] Wilson Abel Alberto Torres, Ron Steinfeld, Amin Sakzad, Joseph K. Liu, Veronika Kuchta, Nandita Bhattacharjee, Man Ho Au, and Jacob Cheng. Post-quantum one-time linkable ring signature and application to ring confidential transactions in blockchain (lattice RingCT v1.0). In Willy Susilo and Guomin Yang, editors, *ACISP 18*, volume 10946 of *LNCS*, pages 558–576. Springer, Heidelberg, July 2018. doi:10.1007/978-3-319-93638-3_32. (Cited on page 153.)

[TU16] Ehsan Ebrahimi Targhi and Dominique Unruh. Post-quantum security of the Fujisaki-Okamoto and OAEP transforms. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B, Part II*, volume 9986 of *LNCS*, pages 192–216. Springer, Heidelberg, October / November 2016. doi:10.1007/978-3-662-53644-5_8. (Cited on page 33.)

[Unr12] Dominique Unruh. Quantum proofs of knowledge. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 135–152. Springer, Heidelberg, April 2012. doi:10.1007/978-3-642-29011-4_10. (Cited on page 152.)

[Unr14] Dominique Unruh. Revocable quantum timed-release encryption. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 129–146. Springer, Heidelberg, May 2014. doi:10.1007/978-3-642-55220-5_8. (Cited on page 155.)

[Unr15a] Dominique Unruh. Computationally binding quantum commitments. Cryptology ePrint Archive, Report 2015/361, 2015. https://eprint.iacr.org/2015/361. (Cited on pages 170 and 171.)

[Unr15b] Dominique Unruh. Non-interactive zero-knowledge proofs in the quantum random oracle model. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 755–784. Springer, Heidelberg, April 2015. doi:10.1007/978-3-662-46803-6_25. (Cited on page 155.)

[Unr15c] Dominique Unruh. Revocable quantum timed-release encryption. *J. ACM*, 62(6):49:1–49:76, 2015. doi:10.1145/2817206. (Cited on pages 32, 153, and 155.)

[Unr16] Dominique Unruh. Computationally binding quantum commitments. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 497–527. Springer, Heidelberg, May 2016. doi:10.1007/978-3-662-49896-5_18. (Cited on page 170.)

[Unr17] Dominique Unruh. Post-quantum security of Fiat-Shamir. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT 2017, Part I*, volume 10624 of *LNCS*, pages 65–95. Springer, Heidelberg, December 2017. doi:10.1007/978-3-319-70694-8_3. (Cited on page 155.)

[Unr21] Dominique Unruh. Compressed permutation oracles (and the collision-resistance of sponge/sha3). *IACR Cryptol. ePrint Arch.*, 2021:62, 2021. URL: https://eprint.iacr.org/2021/062. (Cited on page 155.)

[USS+20] Florian Unterstein, Marc Schink, Thomas Schamberger, Lars Tebelmann, Manuel Ilg, and Johann Heyszl. Retrofitting leakage resilient authenticated encryption to microcontrollers. *IACR TCHES*, 2020(4):365–388, 2020. https://tches.iacr.org/index.php/TCHES/article/view/8687. doi:10.13154/tches.v2020.i4.365-388. (Cited on page 102.)

[Vau16] Serge Vaudenay. Clever arbiters versus malicious adversaries - on the gap between known-input security and chosen-input security. In Peter Y. A. Ryan, David Naccache, and Jean-Jacques Quisquater, editors, *The New Codebreakers - Essays Dedicated to David Kahn on the Occasion of His 85th Birthday*, volume 9100 of *Lecture Notes in Computer Science*, pages 497–517. Springer, 2016. doi:10.1007/978-3-662-49301-4_31. (Cited on page 106.)

[Wat09] John Watrous. Zero-knowledge against quantum attacks. *SIAM J. Comput.*, 39(1):25–58, 2009. URL: https://doi.org/10.1137/060670997. (Cited on pages 152 and 193.)

[WH20] Hongjun Wu and Tao Huang. Tinyjambu. Technical report, National Institute of Standards and Technology, 2020. available at https://csrc.nist.gov/projects/lightweight-cryptography/finalists. (Cited on page 154.)

[Wie83] Stephen Wiesner. Conjugate coding. *SIGACT News*, 15(1):78–88, 1983. doi:10.1145/1008908.1008920. (Cited on page 31.)

[WP16] Hongjun Wu and Bart Preneel. Aegis v1.1. *Submission to the CAESAR Competition*, 2016. (Cited on page 147.)

[Wu16] Hongjun Wu. Acorn v3. *Submission to the CAESAR Competition*, 2016. (Cited on page 147.)

[WZ82] William K Wootters and Wojciech H Zurek. A single quantum cannot be cloned. *Nature*, 299(5886):802–803, 1982. (Cited on pages 31, 152, and 217.)

[Xag13] Keita Xagawa. Message authentication codes secure against additively related-key attacks. Cryptology ePrint Archive, Report 2013/111, 2013. https://eprint.iacr.org/2013/111. (Cited on pages 27, 104, and 105.)

[XAY+20] Haiyang Xue, Man Ho Au, Rupeng Yang, Bei Liang, and Haodong Jiang. Compact authenticated key exchange in the quantum random oracle model. Cryptology ePrint Archive, Report 2020/1282, 2020. https://eprint.iacr.org/2020/1282. (Cited on page 33.)

[Yao82] Andrew Chi-Chih Yao. Protocols for secure computations (extended abstract). In *23rd FOCS*, pages 160–164. IEEE Computer Society Press, November 1982. doi:10.1109/SFCS.1982.38. (Cited on page 179.)

[Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th FOCS*, pages 162–167. IEEE Computer Society Press, October 1986. doi:10.1109/SFCS.1986.25. (Cited on page 179.)

[YDW+12] Guomin Yang, Shanshan Duan, Duncan S. Wong, Chik How Tan, and Huaxiong Wang. Authenticated key exchange under bad randomness. In George Danezis, editor, *FC 2011*, volume 7035 of *LNCS*, pages 113–126. Springer, Heidelberg, February / March 2012. (Cited on page 106.)

[Yil10a] Scott Yilek. *Public-key encryption secure in the presence of randomness failures*. PhD thesis, University of California, San Diego, 2010. (Cited on page 128.)

[Yil10b] Scott Yilek. Resettable public-key encryption: How to encrypt on a virtual machine. In Josef Pieprzyk, editor, *CT-RSA 2010*, volume 5985 of *LNCS*, pages 41–56. Springer, Heidelberg, March 2010. doi:10.1007/978-3-642-11925-5_4. (Cited on pages 28, 29, 104, 105, 106, 128, 129, 134, 136, 138, 139, 140, 144, 147, and 225.)

[YS13] Yu Yu and François-Xavier Standaert. Practical leakage-resilient pseudorandom objects with minimum public randomness. In Ed Dawson, editor, *CT-RSA 2013*, volume 7779 of *LNCS*, pages 223–238. Springer, Heidelberg, February / March 2013. doi:10.1007/978-3-642-36095-4_15. (Cited on page 45.)

[YSPY10] Yu Yu, François-Xavier Standaert, Olivier Pereira, and Moti Yung. Practical leakage-resilient pseudorandom generators. In Ehab Al-Shaer, Angelos D. Keromytis, and Vitaly Shmatikov, editors, *ACM CCS 2010*, pages 141–151. ACM Press, October 2010. doi:10.1145/1866307.1866324. (Cited on page 44.)

[YZ21] Takashi Yamakawa and Mark Zhandry. Classical vs quantum random oracles. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part II*, volume 12697 of *LNCS*, pages 568–597. Springer, Heidelberg, October 2021. doi:10.1007/978-3-030-77886-6_20. (Cited on pages 153 and 155.)

[Zha12a] Mark Zhandry. How to construct quantum random functions. In *53rd FOCS*, pages 679–687. IEEE Computer Society Press, October 2012. doi:10.1109/FOCS.2012.37. (Cited on pages 33, 154, 155, 190, and 195.)

[Zha12b] Mark Zhandry. Secure identity-based encryption in the quantum random oracle model. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 758–775. Springer, Heidelberg, August 2012. doi:10.1007/978-3-642-32009-5_44. (Cited on pages 153, 155, 158, 159, and 216.)

[Zha19a] Mark Zhandry. How to record quantum queries, and applications to quantum indifferentiability. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part II*, volume 11693 of *LNCS*, pages 239–268. Springer,

Heidelberg, August 2019. doi:10.1007/978-3-030-26951-7_9. (Cited on pages 153, 155, 175, and 217.)

[Zha19b] Mark Zhandry. Quantum lightning never strikes the same state twice. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part III*, volume 11478 of *LNCS*, pages 408–438. Springer, Heidelberg, May 2019. doi:10.1007/978-3-030-17659-4_14. (Cited on page 31.)

[Zha20] Jiayu Zhang. Succinct blind quantum computation using a random oracle. Cryptology ePrint Archive, Report 2020/1469, 2020. https://eprint.iacr.org/2020/1469. (Cited on page 33.)

[Zha21] Mark Zhandry. Quantum lightning never strikes the same state twice. or: Quantum money from cryptographic assumptions. *Journal of Cryptology*, 34(1):6, January 2021. doi:10.1007/s00145-020-09372-x. (Cited on page 31.)

[ZRE15] Samee Zahur, Mike Rosulek, and David Evans. Two halves make a whole - reducing data transfer in garbled circuits using half gates. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 220–250. Springer, Heidelberg, April 2015. doi:10.1007/978-3-662-46803-6_8. (Cited on page 187.)

[ZYF+19] Jiang Zhang, Yu Yu, Dengguo Feng, Shuqin Fan, and Zhenfeng Zhang. On the (quantum) random oracle methodology: New separations and more. Cryptology ePrint Archive, Report 2019/1101, 2019. https://eprint.iacr.org/2019/1101. (Cited on page 155.)