Costing Adversaries on Quantum-secure Cryptography

Zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften

von der KIT-Fakultät für Informatik
des Karlsruher Instituts für Technologie (KIT)

genehmigte

Dissertation

von

Kevin Marcel Tiepelt

Tag der mündlichen Prüfung: 23. Januar 2025

1. Referent: Prof. Dr. Jörn Müller-Quade
2. Referent: Prof. Dr. Douglas Stebila
3. Referent: Prof. Dr. Daniel Loebenberger

This version of the dissertation is optimized for digital reading. That means that it is one-sided*, links are embedded into the PDF rather than the margins and the citation in footnotes is increased to one per page (rather than one per a pair of even-odd-pages). The content and page numbers are identical.*

Acknowledgments

The journey to completing my dissertation was challenging, fun, rewarding but also frustrating at times, and would not have been possible without the guidance and support of many incredible individuals and institutions.

First, I want to thank Tilak Singh for introducing me to both academic and industrial research, which essentially set me on the path to pursue a doctoral degree. His encouragement provided my with the foundation for my academic aspirations, but also led me to an amazing experience at Mitsubishi FUSO in Japan.

I am deeply thankful to my supervisor Jörn Müller-Quade and the "Cryptography and Security" research group. Your unwavering support, open-mindedness in discussions and guidance made this a wonderful experience. A special mention to Robin Berger and Michael Klooß for dedicating their time to proofreading parts of my thesis. I am also grateful to Thorsten Strufe for providing guidance on all kinds of meta-level questions that arose throughout this journey, and the "Privacy and Security" group for organizing game-nights and bringing some life to the corridor. Another special mention to Simon Hanisch, for building a wind turbine with me and providing delicious honey (what are you feeding those bees?!).

Thanks to the COSIC research group, particularly Jan-Pieter D'Anvers and Alan Szepiniez for their insight and assistance during my time in Leuven and the early stages of my doctoral studies. Thanks to the Helmholtz Data Science Academy for enabling my work with the German Aerospace Center (DLR), specifically Nils Mäurer, for welcoming me to the team at DLR and fostering collaboration. A special acknowledgement to Michele Mosca, Mitacs and the DAAD for granting me the incredible opportunity to visit (and hosting me at) the Institute for Quantum Computing at the University of Waterloo. That experience significantly enriched my perspective, allowed me to extend my collaboration, network and further ignited my research endeavors. I extend my gratitude to Douglas Stebila and Ted (Edward Eaton) for introducing me to the fascinating world of quantum-annoying'ness, a topic that continues to inspire and shape my ongoing research. To Nina Bindel, Fernando Virdia, and Xavier Bonnetain, thank you for guiding me through the challenging yet rewarding work on lattice enumeration over nearly two years. Despite the many setbacks, the journey was fruitful and transformative. A special mention to Nina for your supervision and support, both in Canada and beyond, which have been invaluable.

A heartfelt thanks to my family and my partner, for your ongoing support and encouragement, not only through this thesis, but throughout my whole life (so far). I cannot thank you enough for standing by me through the highs and lows of my journey. Special mention to Chrystalla Paleshi for constantly helping me to improve my English. A warm thanks to Julian Herr for ongoing technical support and fantastic discussions.

Lastly, I am profoundly grateful to all those who have contributed indirectly (and unknowingly, possibly even unwillingly) to this thesis, such as

everyone who generously shares templates online and to the Stack Exchange community for their ever-helpful advice in resolving LaTeX issues. To all who have walked this journey with me, directly or indirectly, thank you for your support, inspiration, and belief in me.

Costing Adversaries on Quantum-secure Cryptography

Dissertation Thesis

Kevin Marcel Tiepelt

January 2025

Submitted in partial fulfillment of the requirements
for the degree of Doktor der Naturwissenschaften (Dr. rer. nat.)

to the

Department of Informatics
at Karlsruhe Institute of Technology

1st Reviewer	Prof. Dr. Jörn Müller-Quade
2nd Reviewer	Prof. Dr. Douglas Stebila
3rd Reviewer	Prof. Dr. Daniel Loebenberger

Abstract

Technological advancement in the 21st century, such as quantum computers, may pose a significant threat to the security of digital information. At the forefront of security stands cryptography, which is, in 2024, an integral component of almost every electronic device, ensuring the security of information at rest, in use, or in motion.

While quantum computers hold the potential of being able to solve complex problems for science, they also pose a significant risk to the cryptographic schemes in use right now. This is because quantum computing may offer a substantial increase in computational power, having the potential to break cryptographic protocols that are (believed to be) secure when attacked with conventional computers.

Consequently, quantum-secure cryptographic protocols may become essential in the near future, necessitating immediate efforts towards standardization to ensure preparedness. The National Institute of Standards and Technology (NIST) initiated a public competition to identify viable quantum-secure cryptographic algorithms. It is important to recognize, that new cryptographic schemes often come with significant flaws. As such a public evaluation of protocols — following Kerckhoffs's principle, which advocates that the security should solely depend on the secrecy of the key — is essential. The flaws that can be found in cryptographic protocols range from bugs in the implementation, over side-channel attacks to cases where the underlying computationally hard problem is not as difficult as expected. It is necessary, to conduct a thorough analysis of these schemes to identify and address these flaws, ensuring robust and secure cryptographic protocols for the future.

This thesis provides insight into the real-world security of quantum-secure protocols proposed in the NIST post-quantum competition and beyond. Furthermore, we review what security properties can be achieved if *secure* post-quantum schemes exist, and what security guarantees may remain even without any quantum-secure hardness assumption.

Our first contribution is the analysis of three candidates of the NIST competition, their implementations and their underlying hardness assumptions. More specifically, we provide a novel attack that exploits decryption failures in Mersenne-based cryptography, which is applicable to two candidates from the first round of the NIST competition. We provide an implementation of our attack on one of these cryptosystems and show how an attacker can retrieve the secret key. Finally, we identify an attackers abilities to forge universal signatures in the hash-based signature scheme SPHINCS$^+$, which was analyzed as a round three candidate and a finalist to the competition in 2022. We analyze the concrete cost to mount an attack on a fault tolerant quantum computer. Subsequently, we investigate lower bounds on the cost of performing quantum lattice enumeration and report how our findings impact the security of Kyber, a lattice-based finalist in the NIST competition. We support our results with experiments and implementations that allow to reproduce our results, and to apply our methodology to estimate the cost of related cryptographic schemes.

Our second contribution quantifies the security of key agreement protocols against quantum-attackers. We review the security of LDACS, a ground-to-air communication protocol under standardization by the International Civil Aviation Organization. The analysis focuses on the security properties that LDACS can achieve in a post-quantum world with the deployed mutual authenticated key exchange. Particularly, we answer the question which security properties are achieved, when LDACS is instantiated with any post-quantum secure key encapsulation mechanism, for instance, any of the NIST post-quantum finalists. Finally, we explore what security one may achieve in the setting of password authenticated key exchange (PAKE) without a post-quantum hardness assumption, in a world were quantum computing is still expensive. In this setting one can perform a *quantum-annoying* PAKE, which means, that the adversarial cost can be scaled with the size of a password space. We are the first to show how an asymmetric PAKE can be enhanced to provide quantum-annoying security on top of its conventional computational security.

Zusammenfassung

Technische Neuerungen des 21. Jahrhunderts, wie beispielsweise Quantencomputer, stellen eine immer größere Herausforderung für die Sicherheit in der Informationstechnik dar. Eine tragende Rolle spielt dabei *Kryptographie*, welche heutzutage ein integraler Bestandteil von fast jedem elektronischen Produkt ist.

Während Quantencomputer komplexe Probleme in der Wissenschaft lösen könnten, stellen sie auch ein erhebliches Risiko für die derzeit verwendeten kryptografischen Verfahren dar. Das liegt daran, dass mit der neuen Technologie eine bis dahin unerreichte Rechenleistung zur Verfügung stehen könnte, und damit die Gefahr, dass kryptographische Protokolle gebrochen werden.

Folglich könnten Quanten-sichere kryptografische Methoden in naher Zukunft eine wichtige Rollen spielen. Eine Standardisierung solcher Verfahren sollte zu dem Zeitpunkt, dass sie gebraucht werden, bereits abgeschlossen sein. Daher sollten Anstrengungen zur Standardisierung bereits heute angestrebt werden. Das "National Institute of Standards and Technology" (NIST) in den USA führt derzeit einen öffentlichen Wettbewerb mit dem Ziel, praktikable Quanten-sichere kryptografische Algorithmen zu finden. Eine öffentliche Evaluierung von Protokollen — nach dem Kerckhoff'schen Prinzip, das besagt, dass die Sicherheit allein von der Geheimhaltung des Schlüssels abhängt — ist unerlässlich. Die Schwachstellen, die in kryptographischen Protokollen gefunden werden können, reichen von Fehlern in der Implementierung über Seitenkanal-Angriffe bis dahin, dass das zugrundeliegende, komplexitätstheoretische Probleme nicht so schwierig zu lösen ist wie erwartet. Daher ist es ist notwendig, eine gründliche Analyse dieser Verfahren durchzuführen, um diese Schwachstellen zu ermitteln und zu beheben und die Robustheit und Verfügbarkeit sicherer kryptographischer Protokolle für die Zukunft zu gewährleisten.

Diese Arbeit gibt einen Einblick in die (reale) Sicherheit der Quanten-sicheren Protokollen im Rahmen des NIST Post-Quantum-Wettbewerbs, sowie darüber hinaus. Es wird untersucht, welche Sicherheit erreicht werden kann, unter der Annahme, dass Post-Quanten Verfahren existieren, und welche Sicherheitsgarantien auch ohne Quanten-sichere Kryptographie gelten.

Der erste Teil der Ergebnisse dieser Arbeit befassen sich mit der Analyse verschiedener Kandidaten des NIST Post-Quanten Wettbewerbs. Wir stellen einen neuartigen Angriff auf Basis von Entschlüsselungs-Fehlern in Mersenne-basierten Kryptosystemen vor, welche als Kandidaten in der ersten Runde des Wettbewerbs vertreten waren. Weiter analysieren wir die Fähigkeit eines Angreifers beliebige Signaturen des SPHINCS$^+$-Verfahrens zu fälschen. Das SPHINCS$^+$ Signaturverfahren ist einer der Finalisten des NIST Wettbewerbs. Unser dritter Beitrag ist die Analyse von Gitter-Enumeration auf einem Quantencomputer. Mit unserem neuen Algorithmus können die Angreifer-kosten anhand verschiedener Metriken quantifiziert werden, welches eine untere- und obere Abschätzung der Sicherheitsmarge ermöglicht.

Im zweiten Teil dieser Arbeit quantifizieren wir die Sicherheit von Schlüsseltransport-Protokollen gegen Quantenangreifer. Wir überprüfen die Sicherheit von LDACS, einem Boden-Luft-Kommunikationsprotokoll, das derzeit von der Internationalen Zivilluftfahrtorganisation standardisiert wird. Die Analyse konzentriert sich auf die Sicherheitseigenschaften, die LDACS in einer Post-Quantum-Welt mit dem eingesetzten authentifizierten Schlüsselaustausch erreichen kann. Insbesondere beantworten wir die Frage, welche Sicherheit erreicht wird, wenn LDACS mit einem Post-Quantum-Schlüsseltransportverfahren instanziiert wird, zum Beispiel mit einem der den NIST Post-Quantum-Finalisten. Schließlich untersuchen wir, welche Sicherheit man in Passwort-authentifiziertem Schlüsselaustausch erreichen kann, ohne Post-Quantum Annahmen zu treffen, sofern die Anwendung von Quantencomputern kostenintensiv ist.

Publications ·

The following publications are outcomes of my doctoral research.

[BT21] Robin M. Berger and **Marcel Tiepelt**. "On Forging SPHINCS$^+$-Haraka Signatures on a Fault-Tolerant Quantum Computer". In: *Progress in Cryptology - LATINCRYPT 2021 - 7th International Conference on Cryptology and Information Security in Latin America, Bogotá, Colombia, October 6-8, 2021, Proceedings*. Ed. by Patrick Longa and Carla Ràfols. Vol. 12912. Lecture Notes in Computer Science. Springer, 2021, pp. 44–63. DOI: 10.1007/978-3-030-88238-9_3. URL: https://doi.org/10.1007/978-3-030-88238-9_3.

[Bin+24] Nina Bindel, Xavier Bonnetain, **Marcel Tiepelt**, and Fernando Virdia. "Quantum Lattice Enumeration in Limited Depth". In: *Advances in Cryptology – CRYPTO 2024*. Ed. by Leonid Reyzin and Douglas Stebila. Cham: Springer Nature Switzerland, 2024, pp. 72–106. ISBN: 978-3-031-68391-6. DOI: 10.1007/978-3-031-68391-6_3. URL: https://doi.org/10.1007/978-3-031-68391-6_3.

[Boe+21] Franziska Boenisch, Reinhard Munz, **Marcel Tiepelt**, Simon Hanisch, Christiane Kuhn, and Paul Francis. "Side-Channel Attacks on Query-Based Data Anonymization". In: *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*. CCS '21. Virtual Event, Republic of Korea: Association for Computing Machinery, 2021, pp. 1254–1265. ISBN: 9781450384544. DOI: 10.1145/3460120.3484751. URL: https://doi.org/10.1145/3460120.3484751.

[DAn+19] Jan-Pieter D'Anvers, **Marcel Tiepelt**, Frederik Vercauteren, and Ingrid Verbauwhede. "Timing Attacks on Error Correcting Codes in Post-Quantum Schemes". In: *Proceedings of ACM Workshop on Theory of Implementation Security, TIS at CCS 2019, London, UK, November 11, 2019*. Ed. by Begül Bilgin, Svetla Petkova-Nikova, and Vincent Rijmen. ACM, 2019, pp. 2–9. DOI: 10.1145/3338467.3358948. URL: https://doi.org/10.1145/3338467.3358948.

[Mäu+21] Nils Mäurer, Thomas Gräupl, Christoph Gentsch, Tobias Guggemos, **Marcel Tiepelt**, Corinna Schmitt, and Gabi Dreo Rodosek. "A Secure Cell-Attachment Procedure of LDACS". In: *IEEE European Symposium on Security and Privacy Workshops, EuroS&P 2021, Vienna, Austria, September 6-10, 2021*. IEEE, 2021, pp. 113–122. DOI: 10.1109/EuroSPW54576.2021.00019. URL: https://doi.org/10.1109/EuroSPW54576.2021.00019.

[TD20] **Marcel Tiepelt** and Jan-Pieter D'Anvers. "Exploiting Decryption Failures in Mersenne Number Cryptosystems". In: *Proceedings of the 7th on ASIA Public-Key Cryptography Workshop, APKC at AsiaCCS 2020, Taipei, Taiwan, October 6, 2020*. Ed. by Keita Emura and Naoto Yanai. ACM, 2020, pp. 45–54. DOI: 10.1145/3384940.3388957. URL: https://doi.org/10.1145/3384940.3388957.

[TES23] **Marcel Tiepelt**, Edward Eaton, and Douglas Stebila. "Making an Asymmetric PAKE Quantum-Annoying by Hiding Group Elements". In: *Computer Security - ESORICS 2023 - 28th European Symposium on Research in Computer Security, The Hague, The Netherlands, September 25-29, 2023, Proceedings, Part I*. Ed. by Gene Tsudik, Mauro Conti, Kaitai Liang, and Georgios Smaragdakis. Vol. 14344. Lecture Notes in Computer Science. Springer, 2023, pp. 168–188. DOI: 10.1007/978-3-031-50594-2_9. URL: https://doi.org/10.1007/978-3-031-50594-2_9.

[TMM24] **Marcel Tiepelt**, Christian Martin, and Nils Mäurer. "Post-Quantum Ready Key Agreement for Aviation". In: *IACR Communications in Cryptology* 1.1 (Apr. 9, 2024). ISSN: 3006-5496. DOI: 10.62056/aebn2isfg. URL: https://doi.org/10.62056/aebn2isfg.

[TS19] **Marcel Tiepelt** and Alan Szepieniec. "Quantum LLL with an Application to Mersenne Number Cryptosystems". In: *Progress in Cryptology – LATINCRYPT 2019*. Ed. by Peter Schwabe and Nicolas Thériault. Cham: Springer International Publishing, 2019, pp. 3–23. ISBN: 978-3-030-30530-7. DOI: 10.1007/978-3-030-30530-7_1. URL: https://doi.org/10.1007/978-3-030-30530-7_1.

Open-Access Versions

I am grateful to *ePrint*, *arXiv* and the cryptographic community who continuously provide free access to knowledge. The following are the available open access versions of my publications.

[BT21] Robin M. Berger and **Marcel Tiepelt**. *On Forging SPHINCS+-Haraka Signatures on a Fault-tolerant Quantum Computer*. Cryptology ePrint Archive, Paper 2021/1484. https://eprint.iacr.org/2021/1484. 2021. DOI: 10.1007/978-3-030-88238-9_3. URL: https://eprint.iacr.org/2021/1484.

[Bin+23] Nina Bindel, Xavier Bonnetain, **Marcel Tiepelt**, and Fernando Virdia. *Quantum Lattice Enumeration in Limited Depth*. Cryptology ePrint Archive, Paper 2023/1423. https://eprint.iacr.org/2023/1423. 2023. URL: https://eprint.iacr.org/2023/1423.

[DAn+19] Jan-Pieter D'Anvers, **Marcel Tiepelt**, Frederik Vercauteren, and Ingrid Verbauwhede. *Timing attacks on Error Correcting Codes in Post-Quantum Schemes*. Cryptology ePrint Archive, Paper 2019/292. https://eprint.iacr.org/2019/292. 2019. URL: https://eprint.iacr.org/2019/292.

[Mäu+21] Nils Mäurer, Thomas Gräupl, Christoph Gentsch, Tobias Guggemos, **Marcel Tiepelt**, Corinna Schmitt, and Gabi Dreo Rodosek. "A Secure Cell-Attachment Procedure of LDACS". In: *IEEE European Symposium on Security and Privacy Workshops, EuroS&P 2021, Vienna, Austria, September 6-10, 2021*. IEEE, 2021, pp. 113–122. URL: https://elib.dlr.de/142721/1/2021___SRCNAS___A_Secure_Cell_Attachment_Procedure_of_LDACS.pdf.

[TD20] **Marcel Tiepelt** and Jan-Pieter D'Anvers. *Exploiting Decryption Failures in Mersenne Number Cryptosystems*. Cryptology ePrint Archive, Paper 2020/367. https://eprint.iacr.org/2020/367. 2020. DOI: 10.1145/3384940.3388957. URL: https://eprint.iacr.org/2020/367.

[TES23] **Marcel Tiepelt**, Edward Eaton, and Douglas Stebila. *Making an Asymmetric PAKE Quantum-Annoying by Hiding Group Elements*. Cryptology ePrint Archive, Paper 2023/1513. https://eprint.iacr.org/2023/1513. 2023. DOI: 10.1007/978-3-031-50594-2_9. URL: https://eprint.iacr.org/2023/1513.

[TMM24] **Marcel Tiepelt**, Christian Martin, and Nils Maeurer. *Post-Quantum Ready Key Agreement for Aviation*. Cryptology ePrint Archive, Paper 2024/1096. https://eprint.iacr.org/2024/1096. 2024. DOI: 10.62056/aebn2isfg. URL: https://eprint.iacr.org/2024/1096.

[TS19] **Marcel Tiepelt** and Alan Szepieniec. *Quantum LLL with an Application to Mersenne Number Cryptosystems*. Cryptology ePrint Archive, Paper 2019/1027. https://eprint.iacr.org/2019/1027. 2019. DOI: 10.1007/978-3-030-30530-7_1. URL: https://eprint.iacr.org/2019/1027.

Contents

Part I

QUESTIONS & ANSWERS

1

Motivation and Contribution: The Why, the What and the How

1.1 The Why

An Ideal World. Imagine an ideal world where every person, every being, would be able to live a happy, fulfilling, and meaningful life. We could live in harmony with nature, build great cities, or live in peace and quiet on our own. Everyone would be free and could dedicate their mind, spirit and body to whatever brings them joy. Society might be united with a common goal, such as exploring the universe, to discover endless knowledge or to thrive beyond our current realm of understanding. We might also have just found the answer to every question. Now we can sit back, enjoy the pleasure of having achieved everything and watch the stars.

In such an ideal world, there would be no need for *sins*, no greed, wrath, violence, fraud or treachery, (as inspired by [Ali21]). No rules to counter such behavior would need to be agreed on, or enforced, neither by a government nor by any individuals.

[Ali21] Alighieri, *Divine Comedy*

A Real World. But we do not live in an ideal world. Our daily lives are shaped by mistrust and suspicion, even if we are not aware of such feelings; we just believe that we are in a *happy place*. We do not hesitate to inflict harm onto others to enforce our agenda. We lock our doors in fear that someone could enter our homes, we invent sophisticated methods to protect ourselves and everything that we believe is ours, both in the physical and digital world. We certainly do not live in an ideal world...

A Necessary Evil. While we may not live — or be able to build — an ideal world, we strive to create one. Throughout history, humanity has constantly shaped and reshaped the world according to each person's own perception of an ideal world. Our society has come up with rules to facilitate a *somewhat* peaceful community, giving everyone more or less clear explanation and expectations what they can or cannot do to live in the society.

Over the past millennia, humanity has devised a multitude of protective measures, that grant us the illusion of living in an ideal world, and to live a free and happy life within the confines of our society. These measure might be clearly visible, such as the locks that we use on our doors every day, or may be covert, operating beneath the radar of the common observer.

One such covert protective measure is *cryptography*, which is a necessary evil in maintaining the security and stability of our ever expanding digital world. On the one hand side, cryptography is an invaluable tool that enables us to secure our most personal information, but at the same time, imposes a limit to what we can do if we want to rely on the security it provides. It is pervasive in modern society, found on nearly every website, every electronic device and every communication channel. Its use is ubiquitous, from private conversations with loved ones, to ordering cheese-crust pizza. Cryptography presents an precious tool for overcoming mistrust and protecting individuals from those, that fail to follow the rules that are essential to maintaining harmony in our community.

Opportunities and Limitations. Cryptography is not just a protective measure, it is also a fascinating science. Cryptography demonstrates that perfect security is possible, for example, when using a one-time pad. At the same time, that sometimes perfect security is impossible [LC96; May97], or at least not feasible [Ken99] with our current state of technology. That means, to secure most valuable secrets we often have to compromise between *sufficient* security and practicability, as both might not be achievable with cryptography at the same time.

[LC96] Lo and Chau, *Why quantum bit commitment and ideal quantum coin tossing are impossible*

[May97] Mayers, "Unconditionally Secure Quantum Bit Commitment is Impossible"

[Ken99] Kent, "Unconditionally Secure Bit Commitment"

The cryptographers' daily work is to develop methods that allow for the safeguarding of information to ensure the confidentiality, authenticity, and integrity of information, whether it is in motion or at rest. In order to quantify what we consider *sufficient* security, we express a cost to break said security in terms such as time, space, or even tangible expenses such as monetary resources. In doing so, we endeavor to establish security measures against a spectrum of adversaries, each with varying capabilities. The rationale for this quantification is the hope of the existence of mathematical puzzles that can be utilized to secure communication, but which cannot be solved by an attacker in *reasonable* time. In the event that such puzzles indeed exist, it creates the opportunity to establish security relative to the scale and complexity of solving such puzzles.

Asymptotic vs. Concrete Security The protection gained from the hypothetical difficulty of solving such puzzles is commonly refereed to as *asymptotic security*, where an adversaries' resources are quantified relative to the size of the puzzle. Cryptographic schemes proven secure in this model are secure, if the latter is sufficiently large. When we talk about asymptotic security, we commonly use the big-O notation, i. e., $O(n)$, where the O means, roughly, that we ignore any constant and polynomial factors that are not expressed in the variable n. Proving security in such a model is desirable, because it allows us to abstract from a specific machine model or concrete physical implementation to a generic computational model. That means, that security relies on the number of fundamental computational steps relative to the parameter n that are required to compute some algorithm, rather than on the number of CPU cycles for a concrete architecture. This has the benefit that *small* improvements in technology (e. g., a faster multiplication algorithm) do not affect the asymptotic security of a scheme. At the same

time, this might also result in an underestimation of the security, if each of the *fundamental computational steps* already imposes a significant cost.

Security is commonly expressed as bit-security, meaning that if a puzzle provides n bits of security, then any adversary should require $O(2^n)$ resources to solve the puzzle. Constructing the puzzle on the other side should be possible using only $O(n)$ resources. This notion has the benefit of making it relatively simple to compare classes of puzzles that are *equally difficult* to solve. On the other side, asymptotic security may not be satisfactory, unless one is convinced that the factors in cost that we ignored are sufficiently small, or that the puzzle can easily be scaled to a point, where the ignored factors become sufficiently small, but the cost of whoever set-up the puzzle is still small enough.

In concrete security there are commonly three security margins, namely $128, 192$ and 256 bits of security. Taking, for instance, 128 bits, this means that we want any adversary to spend at least 2^{128} *resources* on breaking our puzzles, or otherwise, only succeeds with a very low probability, while setting up the puzzle requires to invest only about 128 resources. This is motivated by the idea that, for example, if an adversary would allocate resources in the form of 2^{128} microseconds of computation to solve the puzzle, they would be busy about 2^{109} years. However, the known universe is only about 2^{34} years old, suggesting that most adversaries will have retired from their evil mission long before solving the puzzle.

Cryptographic schemes are usually proven to be secure in an asymptotic realm, providing security if the underlying puzzle is scaled sufficiently. In order to provide a concrete security the most one can do is to take the algorithm most well-known to humankind to solve the puzzle, and estimate how many resources this algorithm takes. While we can always scale our puzzles to make them more difficult to solve, this incurs a higher cost for the person who sets them up. As such, we want the smallest puzzle that still provides sufficient security. Exploring the gap between the promise from asymptotic security and quantifying the concrete cost an adversary expends to solve such puzzles, thus ensuring that we scale our puzzle sufficiently, brings us one step closer to a real-world that is indistinguishable — at least for an ignorant observer — from an ideal world.

Cryptographic Puzzles. Cryptography has a vast field of applications, the most simple of which are encryption, i. e., locking information into a digital strongbox using a digital key, and authentication, which provides the means to prove that a message send by a certain party was not modified.

In this setting we distinguish between asymmetric or public-key cryptography, and symmetric or secret-key cryptography. In the former, one communication party construct a public-key secret-key pair, publishes the public key and keeps the secret key private. Anyone who has access to the public key can now use this to provide some form of secure communication to the party that has the secret key, for example public-key encryption as in Figure 1.1(a). In secret-key setting, both communication partners need to share the same secret key, for example as in secret-key encryption in Figure 1.1(b). While secret-key cryptography is usually much more efficient in practice,

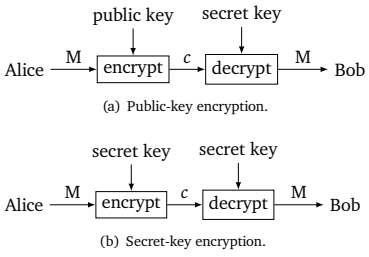


Figure 1.1: Enc- and decryption with plaintext M and ciphertext *c*.

it raises the problem of distributing the keys between the communicating parties, a task that can be achieved securely with asymmetric cryptography.

A *cryptographic puzzle* may then be defined as the difficulty to find this secret key, given only the public key as input. The security of the cryptographic scheme can then be quantified relative to the difficulty to solve this puzzle. With this view on security the underlying cryptographic scheme is secure under the assumption that no one can solve the puzzle efficiently.

1.1.1 *Solving Puzzles*

The computer has undergone a significant development in the past century, starting at room-sized machines that require a complete team of operators, to nail-sized machines that function nearly autonomously. As the size decreased, the computational power increased exponentially for a while, allowing us to solve puzzles that were previously thought to be infeasible to compute for humanity.

The 21st century has even led to the emergence of a potential new type of computer: the quantum computer. At its core, a quantum computer operates very similarly to a conventional computer. It applies an algorithm to a given input and provides an output, or runs for eternity. In contrast to a conventional computer, however, it can take advantage of quantum mechanics, one of the two theories (along with general relativity) that most closely explain our planet, our universe and our very existence. Briefly speaking quantum mechanics explains how individual particles behave and how they evolve over time. It reveals that particles can exists in multiples states at once, called a *superposition*, and how these states interact with each other to become *entangled*. While quantum computing can harness the power that comes with these properties, they are also limited by the very same.

Since the initial conception of quantum computing in the 1980s, technology has come a long way and yet it seems to be a formidable challenge to build such a device that is useful outside of a laboratory. At the time of writing, in 2024, there is no public evidence[1] of a quantum computer that outperforms conventional computers.

A new Threat. While the practical realization of a quantum computer appears to be challenging, the theoretical model has a simple and complete description founded on linear algebra. The model of quantum circuits allows to explore the possibilities promised without having to deal with the problem of constructing a physical computer. This model, which captures the theoretical opportunities and boundaries of quantum computing, allows one to construct algorithms that break widely used encryption methods — implying that quantum computers have the potential to do the same. Quantum computing not only introduces novel attack strategies on cryptographic schemes, but also turns attacks previously considered unfeasible into practical computations thanks to its quantum speedup promise.

In the setting of quantum adversaries, the puzzles used in symmetric cryptography appear largely resistant to quantum attacks, i. e., the best known quantum-speedup in most practical models[2] is at best quadratic, reducing the security margin from n bits to $n/2$ bits [Jaq+20].

[1] In 2019, researchers at Google claimed to have achieved quantum supremacy, after having computed a task where "the equivalent task for a state-of-the-art classical supercomputer would take approximately 10,000 years" [Aru+19]. Shortly after that, still in 2019, IBM published a study which claimed that the "ideal simulation of the same task can be performed on a classical system in 2.5 days and with far greater fidelity" [IBM19]. In 2023, an anonymous author submitted a paper that claimed to have run Google's *quantum supremacy* task on a *Commodore 64* [Ano24], a 40 year old computer.

[2] If an adversary is given quantum access do a decryption oracle some symmetric schemes become vulnerable to Simon's algorithm. These are known as *Superposition* attacks. However, such a security model is highly controversial, if not even questionable.

[Jaq+20] Jaques et al., "Implementing Grover Oracles for Quantum Key Search on AES and LowMC"

On the other side, many of the puzzles used for asymmetric cryptography that are in use today are vulnerable to Shor's algorithm [Sho94; GE21] which can solve these puzzles exponentially faster than a classical computer. If somebody were to build a scalable, fault-tolerant quantum computer, this would mean that currently used cryptography would be insecure, and scaling the puzzle size to reach the desired security would make the schemes unpractical, for example, resulting in key-sizes in the magnitude of terabytes [Ber+17].

[Sho94] Shor, "Algorithms for quantum computation: discrete logarithms and factoring"

[GE21] Gidney and Ekerå, "How to factor 2048 bit RSA integers in 8 hours using 20 million noisy qubits"

[Ber+17] Bernstein et al., *Post-quantum RSA*

Quantum-secure Cryptography The need to develop new, quantum-secure cryptographic protocols has been recognized by several standardization bodies such as the German Federal Office for Information Security (BSI) and the National Institute for Standards and Technology (NIST) in the United States of America, the latter of which started a post-quantum standardization process [Nat17] to select one or more post-quantum replacements for the current public-key cryptographic primitives. These replacements are based on computational puzzles, that are believed to be difficult to solve even for quantum computers – and which can be used to build secure cryptographic protocols.

[Nat17] National Institute for Standards and Technology, *Post-Quantum Cryptography Call for Proposals*

The adoption of quantum secure cryptographic schemes comes with formidable challenges, as this necessitates the overhaul of existing protocols and the assessment of the security provided by quantum-secure intractability assumptions, i. e., assumptions that a certain puzzle is difficult to solve even with quantum computers. The trust in these assumptions often comes from reductions to other, more general puzzles or computational problems. Say we have two puzzles, A and B. If one can show that an algorithm that solves puzzle A can be used to build an algorithm to solve puzzle B, then solving B cannot be more difficult than solving puzzle A. If one can now provide evidence that puzzle B is *difficult* to solve, then one can also assume that A is difficult to solve.

In the setting of computational complexity, quantum computers can solve a special set of puzzles located in a class called Bounded Error Quantum Polynomial Time (BQP) as in Figure 1.2. These problems are defined as decision problems, meaning that solving the puzzle requires an algorithm to output "yes" or "no". The decision problems that conventional deterministic computers can solve, that coincide with the algorithm that ought to be used for encryption and decryption, can be thought of as lying in the class "P". The problems in the subset "NP-complete" of the class "NP" are believed to be intractable even for quantum computers. However, it is not known[3] if they can be utilized for cryptography. As such, quantum secure cryptography is based on puzzles located somewhere "in between", i. e., puzzles that are *close* to an NP-complete problem, but that have been artificially made *easier*. We call these puzzles *intractability assumptions* or *hardness assumptions*.

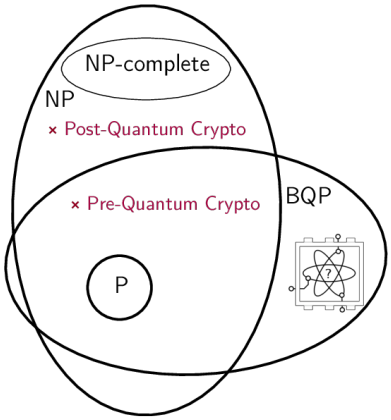


Figure 1.2: Placement of the power of quantum computing "BQP" in complexity theory. "Post-Quantum Crypto" refers to quantum-secure, "Pre-Quantum Crypto" to the conventional secure cryptosystems.

[3] Proving that a cryptographic scheme is as difficult to break as solving an NP-compete problem implies that NP = co-NP [Bra83, Thm. 2], which is among the biggest open questions in computer science.

1.2 THE WHAT — RESEARCH QUESTIONS

The concrete difficulty of solving puzzles is often not clear. Embedding these puzzles into existing cryptographic protocol or constructing new protocols often entail modifications to cryptographic primitives, which potentially also

introduces new vulnerabilities. As such, there may be a large gap between the asymptotic hardness of intractability assumptions, and the concrete security a system provides.

To be confident in a cryptographic protocol we need to be assured, that it is as least as difficult to break as the intractability assumption, and that "breaking" this assumption is indeed computationally hard. For the latter, we want to know:

> *How difficult is it to solve a computational problem?*

For many cases the answer is simple — we do not know for sure. Instead, the difficulty of solving a problem has been subject to years or decades of research, trying to find an *efficient* algorithm to solve the specific problem. With the tools from complexity theory, i. e., reductions to problems of *similar* difficulty, the reasoning that no efficient algorithm exists for a specific problem can be extended to providing evidence that no efficient algorithm exists to solve other (possibly more generic) problems.

In a nutshell, the security of classical cryptographic schemes is based on the assumption that the most well known algorithms require an exponential amount of resources to solve the problems in question, and it is widely accepted throughout the cryptographic community that 128-bits of security are sufficient. With quantum computing emerging, some of these assumptions have been challenged. Not only do quantum algorithms have the potential to break well-studied assumptions, but they also enable attacks which formerly believed to be unpractical. This leads us to our first research question:

> *Q1 — What are the* best *classical-quantum attack strategies on post-quantum cryptography?*

The "best" algorithm is often quantified to be the one with the smallest cost. As such, it is equally important to ask:

> *Q2 — What is the concrete cost of the* best *classical-quantum attacks?*

While the asymptotic behavior of quantum computers has been extensively analyzed, the possibilities within these limits, and the practical embodiment remain ambiguous. Naturally, this results in a gap between the asymptotic cost to break the intractability assumption and the difficulty to break the cryptographic scheme in practice. Therefore it is necessary to understand the ability of quantum-algorithms to solve these assumptions. The goal is to show that the "most well-known algorithm" requires 2^X resources in a computational model as depicted in Figure 1.3. A scheme would than be considered secure, if 2^X is larger than some practical bound, for example, larger than 2^{128}.

Once we are convinced of the security of a cryptographic scheme, the next challenge is to embed the new cryptographic protocol into existing, higher-level protocols, for example, key agreement schemes. This requires demonstrating that certain security properties hold, for example, then we can show that peers can be convinced to share a secret key, or convinced

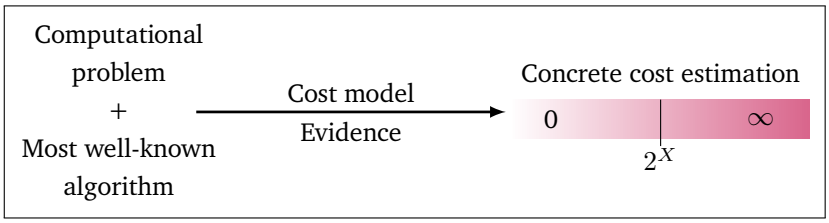


FIGURE 1.3: The concrete cost of solving a computational problem using the "most well-known algorithm" is shown in a specific cost model and providing some evidence, for example, an implementation or experiments.

that they are communicating with their intended peer. Conversely, if the cost of quantum computation is prohibitively high, one may ask whether existing components and schemes that do not provide security against arbitrarily large quantum computers could be enhanced to make the cost of the attacker prohibitive once more. This leads to the final research question:

> Q3 — What security can we achieve against *limited* quantum adversaries?

A limitation on an adversary is potentially imposed on the cost model by current technology, for example, by considering only quantum circuits that remain coherent for a limited time, or by allowing the adversary only specific powers. In the following we review how these questions are addressed in this manuscript.

1.3 THE HOW — RESULTS AND PUBLICATIONS

In the remaining of Part I, specifically Chapter 2, we introduce standard notation used throughout this manuscript. Then the research questions are addressed in two distinct settings: In Part II, we quantify the cost of adversaries on three candidates of the NIST post-quantum competition. For each candidate we provide either an attack on the scheme, or evidence, suggesting that the intractability assumptions are difficult to solve even for quantum computers. Conversely, in Part III, we review the security that can be achieved in higher level protocols under the assumption that post-quantum secure cryptographic primitives exist. We then explore a security level that can be achieved by strengthening the quantum-security of cryptographic protocols, that would otherwise be secure against classical computation only.

1.3.1 *Part II: Costing Adversaries on Post-Quantum Cryptography*

The most prominent quantum-secure cryptosystems are the candidates of the NIST post-quantum competition [Nat17]. We review the security of the Mersenne-based Key Encapsulation Mechanism (KEM) Ramstake from Round 1, the Hash-based signatures schemes SPHINCS$^+$ that was chosen as a finalist, and the security of the lattice-based finalist Kyber. Table 1.1 summarizes our contributions for each of the candidates.

[Nat17] National Institute for Standards and Technology, *Post-Quantum Cryptography Call for Proposals*

Before evaluating the security of specific schemes we devote Chapter 3 to a review of cost models and algorithms in the setting of quantum computing. A review of the individual cryptosystems is provided in Chapter 4,

accompanied by an examination of the most well-known attacks on each cryptographic scheme. Subsequently, we present our analysis of the cryptosystems in question and of their underlying hardness assumptions. For each such scheme, we either provide evidence that the computational problem corresponding to the respective assumption is difficult to solve in practice — or show how the scheme can be broken *efficiently*.

TABLE 1.1: Overview of contributions to the public analysis of NIST post-quantum schemes.

Candidate	Impact/ Results	Evidence
Ramstake (1st round)	IND-CCA security broken + secret-key extraction	Theoretical analysis, implementation of attack
SPHINCS$^+$ (3rd round, finalist)	Improved 2^{nd} preimage attack + fault-tolerant cost estimation	Theoretical analysis, quantum circuit simulation
Kyber (finalist)	New quantum lattice enumeration algorithm + instantiation barriers	Theoretical analysis, heuristic experiments, optimal cost-estimation

CHAPTER 5. Mersenne-based cryptosystems were first introduced in 2017 [Agg+17a] as single bit encryption scheme and later submitted to the NIST competition as the Mersenne-756839 [Agg+17b] and the Ramstake [Sze17] key encapsulation mechanism. The two submissions have a small, but non-zero probability that a decryption fails, and thus that no key is exchanged.

We develop the, to the best of our knowledge, first method to exploit these decryption failures in Mersenne number based cryptosystems. These encryption schemes combine arithmetic modulo a Mersenne prime with error correcting codes to form a key encapsulation mechanism. The incorporation of the error correcting codes results in a low, but non-zero, probability that the decryption fails. In this chapter we show that the information leaked from such a decryption failure can be used to gain information about the secret key. We present an attack exploiting this information to break the Indistinguishability under Chosen Ciphertext Attack (IND-CCA) security of Ramstake [Sze17], a round one candidate in the NIST post-quantum competition.
Particularly, our method takes a set of decryption failures and yields a probability distribution of the secrets. The distribution can be used to significantly improve the complexity of the most well-known algorithm to solve the underlying assumption. This results in an attack that not only breaks the IND-CCA security, but also extracts the secret keys.

In this collaboration, Jan-Pieter D'Anvers mainly developed a method based on maximum likelihood estimation to output a distribution of the secret bit-positions from decryption failures. My contribution was to develop a method that extracts a set of intervals that contain a *one* from this distribution, and to adapt as well as improve the performance of the most well-known attack using these intervals, enabling to apply a lattice-reduction aided attack.

[Agg+17a] Aggarwal et al., *A New Public-Key Cryptosystem via Mersenne Numbers*

[Agg+17b] Aggarwal et al., *Mersenne-756839*

[Sze17] Szepieniec, *Ramstake*

We demonstrate the efficacy of our attack by providing an implementation on a simplified version of the candidate *Ramstake* [Sze17].

[Sze17] Szepieniec, *Ramstake*

Content Sources

Chapter 5 is based on below publications.

Publication

Marcel Tiepelt and Jan-Pieter D'Anvers. "Exploiting Decryption Failures in Mersenne Number Cryptosystems". In: *Proceedings of the 7th on ASIA Public-Key Cryptography Workshop, APKC at AsiaCCS 2020, Taipei, Taiwan, October 6, 2020*. Ed. by Keita Emura and Naoto Yanai. ACM, 2020, pp. 45–54. DOI: 10.1145/3384940.3388957

Open-Access Publication

Marcel Tiepelt and Jan-Pieter D'Anvers. *Exploiting Decryption Failures in Mersenne Number Cryptosystems*. Cryptology ePrint Archive, Paper 2020/367. https://eprint.iacr.org/2020/367. 2020. DOI: 10.1145/3384940.3388957

Implementation

https://github.com/mtiepelt/ramstake-failure-attack{}

Contribution Equal.

Chapter 6. The SPHINCS$^+$ signature scheme [Hül+20] has been chosen by the NIST as one of finalists of the post quantum competition [Nat22], and standardized in August 2024 [Nat24c]. The security of the scheme is based on the second-preimage resistance of cryptographic hash functions, namely Haraka, SHA-256 or SHAKE-256. While the cost of computing a generic preimage of SHA-256 and SHAKE-256 has been analyzed before [Amy+16], the cost of attacking Haraka remains unknown. The SPHINCS$^+$ signature scheme is constructed from three distinct signatures schemes: A Winternitz One Time Signature (WOTS), a Forest of Random Subsets (FORS) and a eXtended Merle Signature Scheme (XMSS), which are combined in a large hypertree. A concrete algorithm that allows to compute a universal forgery for SPHINCS$^+$ remains ambiguous.

[Hül+20] Hülsing et al., *SPHINCS+-Submission to the 3rd round of the NIST post-quantum project*

[Nat22] National Institute for Standards and Technology, *NIST: Selected Algorithms 2022*

[Nat24c] National Institute for Standards and Technology, *FIPS 205*

[Amy+16] Amy et al., "Estimating the Cost of Generic Quantum Pre-image Attacks on SHA-2 and SHA-3"

We investigate the cost of performing a universal forgery on the signatures from a second preimage in the underlying hash function, and provide the first analysis of SPHINCS$^+$-Haraka relative to an adversaries' ability to find a second preimage in the hash function using quantum amplitude amplification.

In each individual component, the best point of attack relative to the cost of the oracle required for quantum amplitude amplification is identified. Subsequently, we review the concrete cost of performing the attack with the lowest cost on a fault-tolerant quantum computer, focusing on the SPHINCS$^+$ instantiation that deploys the Haraka hash function. To that end, we use a Q# implementation that was developed with Robin Berger in his Master's Thesis as a baseline for the cost of a quantum oracle of the Haraka hash function. A detailed cost analysis is then conducted, that takes into consideration the overhead of turning faulty quantum gates and quantum bits into fault-tolerant logical resources. As a result we obtain a cost estimation of

performing a universal forgery of SPHINCS$^+$ on a fault-tolerant quantum computer, which improves over the cost estimation from previous, generic attacks on the scheme.

In this collaboration, my contribution was development of the explicit universal forgery attacks on the SPHINCS$^+$ components in close cooperation with Robin Berger. For the resource estimation Robin developed most of the Q# code to perform the logical cost analysis, while my contribution focused on the optimization in the fault-tolerant regime.

Content Sources

Chapter 6 is based on below publications.

Publication

Robin M. Berger and **Marcel Tiepelt**. "On Forging SPHINCS$^+$-Haraka Signatures on a Fault-Tolerant Quantum Computer". In: *Progress in Cryptology - LATINCRYPT 2021 - 7th International Conference on Cryptology and Information Security in Latin America, Bogotá, Colombia, October 6-8, 2021, Proceedings*. Ed. by Patrick Longa and Carla Ràfols. Vol. 12912. Lecture Notes in Computer Science. Springer, 2021, pp. 44–63. DOI: 10.1007/978-3-030-88238-9_3

Open-Access Publication

Robin M. Berger and **Marcel Tiepelt**. *On Forging SPHINCS+-Haraka Signatures on a Fault-tolerant Quantum Computer*. Cryptology ePrint Archive, Paper 2021/1484. https://eprint.iacr.org/2021/1484. 2021. DOI: 10.1007/978-3-030-88238-9_3

Implementation

https://github.com/RobinBerger/Grover-Sphincs{}

Contribution Equal.

CHAPTER 7. Three out of four of the candidates selected for standardization by NIST [Nat22] are lattice-based construction, two of which were standardized in August 2024 [Nat24a; Nat24b]. The most well-known attacks on lattice-based cryptosystems are either based on lattice enumeration, or lattice sieving. The quantum variant of the latter has been proposed [LMv13], but also been shown to not be competitive [Alb+20b]. To speedup lattice enumeration Aono, Nguyen, and Shen proposed to deploy quantum backtracking algorithms [Mon18; AK17] in an unbounded quantum circuit model. We explore the cost of quantum lattice enumeration in the setting of NIST's MAXDEPTH, i. e., the limitation of the number of consecutive gates in any quantum circuits, and propose a parallelization strategy, resulting in a new classical-quantum enumeration algorithm. We detail the cost based on various assumption and conjectures, which aim to find *meaningful* lower bounds under these restrictions. We apply our estimates to Kyber, a finalist in the NIST post-quantum competition.

In order to argue that our bounds are meaningful, Fernando Virdia provided a large amount of lattice-based heuristics, which will not be part of this thesis. My contribution was, in close collaboration, the design and analysis of the classical-quantum algorithm and the implementation and optimization of the resources estimate. Our figures and tables can be reproduced with our public code.

[Nat22] National Institute for Standards and Technology, *NIST: Selected Algorithms 2022*

[Nat24a] National Institute for Standards and Technology, *FIPS 203*

[Nat24b] National Institute for Standards and Technology, *FIPS 204*

[LMv13] Laarhoven, Mosca, and van de Pol, "Solving the Shortest Vector Problem in Lattices Faster Using Quantum Search"

[Alb+20b] Albrecht et al., "Estimating Quantum Speedups for Lattice Sieves"

[Mon18] Montanaro, "Quantum-Walk Speedup of Backtracking Algorithms"

[AK17] Ambainis and Kokainis, "Quantum algorithm for tree size estimation, with applications to backtracking and 2-player games"

Content Sources

Chapter 7 is based on below publications.

Publication

Nina Bindel, Xavier Bonnetain, **Marcel Tiepelt**, and Fernando Virdia. "Quantum Lattice Enumeration in Limited Depth". In: *Advances in Cryptology – CRYPTO 2024*. Ed. by Leonid Reyzin and Douglas Stebila. Cham: Springer Nature Switzerland, 2024, pp. 72–106. ISBN: 978-3-031-68391-6. DOI: 10.1007/978-3-031-68391-6_3

Open-Access Publication

Nina Bindel, Xavier Bonnetain, **Marcel Tiepelt**, and Fernando Virdia. *Quantum Lattice Enumeration in Limited Depth*. Cryptology ePrint Archive, Paper 2023/1423. https://eprint.iacr.org/2023/1423. 2023

Implementation

https://github.com/mtiepelt/QuantumLatticeEnumeration

Contribution Equal.

1.3.2 *Part III: Quantum-secure Protocols*

In the previous Part II we address the question what the concrete cost of an attacker with a quantum computer is, and it turns out, running quantum-algorithms in practice appears to be very expensive. This raises the question what cost asymmetric cryptography, which is based on assumptions that can be solved in polynomial time in the quantum regime, incurs to an attacker. Indeed, if quantum computation is very expensive, schemes based on such an assumption may still provide some security, and may be further strengthened by providing the ability to scale the computationally cost specifically for quantum computers.

On the other side, even if post-quantum secure primitives exists and the cost is well understood, these can often not be readily put into key agreement schemes. Indeed, asymmetric cryptographic protocols are often not deployed *directly*. Instead, key encapsulation mechanisms, public-key encryption and signature protocols are embedded into higher level key agreement schemes that combine these primitives to achieve, for example, *authenticated key exchange*. If protocols are now based on new primitives not used before, security proofs may be invalidated. As such, when instantiating a higher level protocol with a post-quantum scheme, it may not be clear what security properties remain. In Chapter 8, we first review a security model that allows us to analyze the security of higher level protocols. Subsequently, we analyze what security guarantees remains when plugging post-quantum security into a higher level protocol, and finally what security can be achieved without post-quantum cryptography.

CHAPTER 9. The upcoming air-to-ground communication system, LDACS [SES23], is currently under standardization by the International Civil Aviation Organization. As the system is to remain secure for the coming decades, the communication must provide security against potential quantum threats. As part of the standardization, the system deploys a key-encapsulation based

[SES23] SESAR JU, *LDACS A/G Specification, Edition 01.01.00, Template Edition 02.00.05, Edition date 25.04.2023*

variant of the *ISO Key Agreement Mechanism 7*. While the protocol has been subject to *heuristic* security reviews, no formal treatment of the properties attained has been conducted before. Therefore, we provide the first formal analysis of the underlying the key agreements protocol and prove that it achieves various desired security properties against quantum adversaries in both, a computational and a symbolic model.

My contribution in this collaboration was the setup and adjusting the formal model. The security proof was conducted in close collaboration with Christian Martin. Large parts of the proof of BR-Secrecy are consistent with the results from his Master's thesis, which are omitted from this manuscript.

Content Sources

Chapter 9 is based on below publications.

Publication

Marcel Tiepelt, Christian Martin, and Nils Mäurer. "Post-Quantum Ready Key Agreement for Aviation". In: *IACR Communications in Cryptology* 1.1 (Apr. 9, 2024). ISSN: 3006-5496. DOI: 10.62056/aebn2isfg

Open-Access Publication

Marcel Tiepelt, Christian Martin, and Nils Maeurer. *Post-Quantum Ready Key Agreement for Aviation*. Cryptology ePrint Archive, Paper 2024/1096. https://eprint.iacr.org/2024/1096. 2024. DOI: 10.62056/aebn2isfg

Implementation

https://github.com/mtiepelt/ldacs-make-symbolic-tamarin

Contribution Equal.

CHAPTER 10. Password Authenticated Key Exchange (PAKE) allows a client to authenticate towards a server using a password, without revealing their password to either the server or the network. A PAKE protocol promises that an attacker is reduced to either solving a computational problem, or performing an online interaction for every password guess. However, the most efficient PAKEs are based on the discrete logarithm problem which is susceptible to quantum attacks, allowing an adversary to find the password after solving a single discrete logarithm. In this work we show how to augment a password authenticated key exchange protocol using an ideal cipher to make the protocol quantum annoying — meaning that we can quantify adversaries probability to find the password based on the number of discrete logarithms they solve — namely, one for every password guess. This significantly increases the adversarial cost as long as quantum computation remains costly. My contribution to this chapter was the augmentation of the password based protocol, and the design and proof of the quantum-annoying property.

Content Sources

Chapter 10 is based on below publications.

Publication

Marcel Tiepelt, Edward Eaton, and Douglas Stebila. "Making an Asymmetric PAKE Quantum-Annoying by Hiding Group Elements". In: *Computer Security - ESORICS 2023 - 28th European Symposium on Research in Computer Security, The Hague, The Netherlands, September 25-29, 2023, Proceedings, Part I*. ed. by Gene Tsudik, Mauro Conti, Kaitai Liang, and Georgios Smaragdakis. Vol. 14344. Lecture Notes in Computer Science. Springer, 2023, pp. 168–188. DOI: 10.1007/978-3-031-50594-2_9

Open-Access Publication

Marcel Tiepelt, Edward Eaton, and Douglas Stebila. *Making an Asymmetric PAKE Quantum-Annoying by Hiding Group Elements*. Cryptology ePrint Archive, Paper 2023/1513. https://eprint.iacr.org/2023/1513. 2023. DOI: 10.1007/978-3-031-50594-2_9

Contribution Major.

1.4 OTHER HOWS — OTHER PUBLICATIONS

The following publications are results of collaborations during the past years which are not part of this thesis, but mentioned here in the interest of completeness.

TIMING ATTACK ON ERROR CORRECTING CODES. We developed timing attacks on error correcting codes of lattice- and Mersenne-based implementations of NIST post-quantum candidates, and demonstrate how the resulting information can be used to break the IND-CCA security of the underlying schemes. My contribution in this paper was an attack on Ramstake, a round-1 Mersenne-based cryptosystem. The attack can extract bit positions of the secret keys based on timing variations of the deployed error correcting code. A complete extraction of the secret is possible within minutes.

- Jan-Pieter D'Anvers, Marcel Tiepelt, Frederik Vercauteren, and Ingrid Verbauwhede. "Timing Attacks on Error Correcting Codes in Post-Quantum Schemes". In: *Proceedings of ACM Workshop on Theory of Implementation Security, TIS at CCS 2019, London, UK, November 11, 2019*. Ed. by Begül Bilgin, Svetla Petkova-Nikova, and Vincent Rijmen. ACM, 2019, pp. 2–9. DOI: 10.1145/3338467.3358948

TIMING ATTACK ON PRIVACY DATABASES. We present a timing attack on privacy databases, particularly, the Diffix database. My contribution was the development and implementation of the timing attack against the Diffix proxy, a privacy preserving database with the promise not to reveal information about users when queried with SQL statements. We exploit that Diffix deployed different methods to retrieve data depending on the SQL query returning some data (slower) or none at all (faster). The cause for this was discovered only after our attack, which allowed to recover the identity of users in the database within a few minutes and using a few thousand queries per user [4].

[4] *Disclosure:* We received a bounty for our attack as part of the *aircloak* bounty challenge in 2020. Unfortunately, the corresponding webpage is now offline, making it impossible to reference it.

- Franziska Boenisch, Reinhard Munz, Marcel Tiepelt, Simon Hanisch, Christiane Kuhn, and Paul Francis. "Side-Channel Attacks on Query-Based Data Anonymization". In: *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*. CCS '21. Virtual Event, Republic of Korea: Association for Computing Machinery, 2021, pp. 1254–1265. ISBN: 9781450384544. DOI: 10.1145/3460120.3484751

QUANTUM LLL. In this paper we offer a design of the famous Lenstra-Lenstra-Lovász (LLL) lattice reduction algorithm as a quantum circuit, providing an optimized un-computation schedule, thus minimizing the memory overhead. We apply our results by estimating the cost of lattice-reduction attacks on Mersenne-based cryptosystem by giving an upper bound on the Toffoli gates required to implement the attack. My contribution was modelling and optimizing the LLL algorithm as a quantum circuit, and estimating the quantum cost of the Slice-and-Dice attack (cf. Section 4.1.3).

- Marcel Tiepelt and Alan Szepieniec. "Quantum LLL with an Application to Mersenne Number Cryptosystems". In: *Progress in Cryptology – LATINCRYPT 2019*. Ed. by Peter Schwabe and Nicolas Thériault. Cham: Springer International Publishing, 2019, pp. 3–23. ISBN: 978-3-030-30530-7. DOI: 10.1007/978-3-030-30530-7_1[5]

[5] While this work would fit well into this manuscript is has some overlap with my Master's thesis, making it unsuited to be part of the dissertation.

POST-QUANTUM CRYPTOGRAPHY FOR LDACS. In collaboration with Institute for Communications and Navigation at the German Aerospace Center we design and tailored a mutual authenticated post-quantum key agreements protocol to constraint resource environments, i. e., the LDACS [SES23] protocol for commercial ground-air communication. My contribution was to design the authenticated key agreement in close collaboration with Nils Mäurer.

[SES23] SESAR JU, *LDACS A/G Specification, Edition 01.01.00, Template Edition 02.00.05, Edition date 25.04.2023*

- Nils Mäurer, Thomas Gräupl, Christoph Gentsch, Tobias Guggemos, Marcel Tiepelt, Corinna Schmitt, and Gabi Dreo Rodosek. "A Secure Cell-Attachment Procedure of LDACS". in: *IEEE European Symposium on Security and Privacy Workshops, EuroS&P 2021, Vienna, Austria, September 6-10, 2021*. IEEE, 2021, pp. 113–122. DOI: 10.1109/EuroSPW54576.2021.00019

2

Foundations

2.1 NOTATION

Parts of this chapter have been taken verbatim from our publications, i. e., [TD20a; TD20b; BT21a; BT21b; Bin+24; Bin+23; TES23b; TES23a; TMM24b; TMM24a].

Throughout this manuscript variables and algorithms are defined at a local level, while maintaining consistent notation across all chapters. This chapter is devoted to introducing notation as well as cryptographic core components.

Sets $\{\}$ is an unordered set, [] an ordered set. $[x]$ are the numbers from 1 to x inclusive.

Strings $\{0,1\}^n$ is a distribution of n-bit strings of the set $\{0,1\}$. If x is a string, then $|x|$ denotes the length of the string. $[x : y]$ denotes a substring, including x, excluding y, corresponding to $[x, y)$ in set notation.

Vectors $\vec{v} = (v_1, v_2, ..., v_n)$ is a n-dimensional vector with coefficients v_i. $||v||$ is the l_2-norm of a vector calculated as $\sqrt{\sum_i v_i^2}$.

Algebra $\mathbb{Z}$ is the set of all integers, $\mathbb{N}$ the natural numbers and $\mathbb{R}$ respectively all real numbers. $\mathbb{Z}_q$ is the ring modulo q. For any integer $x \in \mathbb{Z}_p$ the i^{th} bit of the binary representation will be expressed as $x[i]$ or in shorthand x_i, and the bits in the range from i to, but not including j, will be written as $x[i : j]$. The integers in this ring will sometimes be expressed as a binary string, using the least significant bit representation in $[0, p)$.

Algorithms and Functions Algorithms are denoted as capital letters, i. e., ALGORITHM. $y \leftarrow f(x)$ assigns the value of $f(x)$ to y. The function $x \xleftarrow{\$} D$ describes the process of drawing uniformly random values from the distribution D. $x := y$ means we define x to be y.

Stochastic $\mathbb{E}[X]$ is the expected value of the random variable X. $\mathbb{P}[E]$ is the probability of the event E happening.

Quantum $\langle\phi||\phi\rangle$ is the bra-ket notation. Quantum states are denoted as lower case Greek letters $|\phi\rangle, |\psi\rangle, ...$ Registers are upper case letters $|A\rangle$, single qubits as lower case letters $|a\rangle$. Amplitudes are also denoted as Greek letters α, β.

Variables $Z.a$ means a is part of Z. For example, if $Z = (a, b, c)$ then we may write *Z.a*, *Z.b* or *Z.c*.

Cryptography λ is the security parameter, A is the adversary and A^f means that the adversary gets oracle access to a function f. *Adv* is the advantage that an adversary has to win a security game G.

ASYMPTOTIC NOTATION. In the setting of cryptography the *cost* of computing a function in usually expressed relative to the length of the input. This cost is then set into relation to the security parameter λ. We use the following asymptotic notation for function $f : \mathbb{N} \to \mathbb{R}^+$:

$f(x) \in$ ***negl*** We call a function *negligible*, if for every $a \in \mathbb{N}$ there exists integer y such that $f(x) \leq x^{-a}$ for all $x \geq y$, i. e., it approaches zero faster than the inverse of any polynomial in the security parameter λ.

$f(x) \in$ ***poly(x)*** means bounded by a polynomial in x.

$f(x) \in o(g(x))$ A function f grows smaller than g if: $\forall c > 0\ \exists x_0\ \forall x > x_0$ $c \cdot |g(x)| > |f(x)|$.

$f(x) \in O(g(x))$ The function $g(x)$ is an asymptotic upper bound if $\exists c > 0$ $\exists x_0 > 0\ \forall x > x_0\ |f(x)| < c \cdot |g(x)|$.

$f(x) \in \Omega(g(x))$ The function $g(x)$ is an asymptotic lower bound, if $\exists c > 0$ $\forall x_0\ \exists x > x_0\ c \cdot |g(x)| \leq |f(x)|$.

$f(x) \in \Theta(g(x))$ The function $g(x)$ is an asymptotic lower and upper bound, if $f(x) \in \Theta(g(x)) \Leftrightarrow f(x) \in O(g(x)) \wedge f(x) \in \Omega(g(x))$.

$\tilde{X}$ This means that for X all logarithmic factors are ignored, for example $\tilde{O}$.

2.2 CRYPTOGRAPHIC COMPONENTS AND SECURITY NOTIONS

Cryptographic components are algorithms that are designed to achieve a security goal. The goal is defined by a security notion associated with a *game* that is played between a challenger and an adversary *A*. The cryptographic component is then secure, if the probability of the adversary winning the game is *sufficiently* small. In the context of *indistinguishability* games, where the adversary wins the game if they make a correct guess, the security is often defined over their advantage *Adv* over guessing.

Security notions capture desired proprieties, security guarantees and the adversarial model, thereby also defining what constitutes a violation of the security of a cryptographic scheme. Consequently, a security proof does not guarantee security against *any* real-world adversary. Instead, the security is limited to the specific adversarial model associated with the security notion.

In this manuscript we generally consider adversaries with access to a quantum computer. This can either be a fully fault-tolerant quantum computer, with access to large, scalable quantum circuits, or an adversary that is limited in their power. Such limitations may be expressed, for example, by limiting the depth of the quantum circuits that can perform a coherent computation as in Section 3.4, or a restriction on the specific unitary they can execute.

In the setting of post-quantum cryptography, cryptographic protocols are proven to fulfill a security notion by reduction to a quantum-computationally

TABLE 2.1: Overview of the notation used for algorithms throughout this manuscript.

Algorithms	Context
$c, k \leftarrow \textsc{Encaps}(pk)$	Public-key encapsulation using public key pk produces a ciphertext c and key k.
$\{k, \perp\} \leftarrow \textsc{Decaps}(sk, c)$	Decapsulation of the ciphertext c using the secret key sk produces a key k or $\perp$.
$c \leftarrow \textsc{Encrypt}(pk, M)$	Public-key encryption of message M under public key pk produces a ciphertext c.
$M \leftarrow \textsc{Decrypt}(sk, c)$	Decryption of the ciphertext using the secret key sk produces the plaintext M.
$\sigma \leftarrow \textsc{Sign}(sk, M)$	Signature under the secret key sk of message M produces the signature σ.
$\{0, 1\} \leftarrow \textsc{Verify}(vk, \sigma, M)$	Public-key verification of the signature σ and message M under the public key vk outputs either 1 if the verification was successful or 0 otherwise.
$\tau \leftarrow \text{MAC}(sk, M)$	Generation of message authentication tag τ under the secret key sk of message M.

hard assumption, i. e., an NP-problem that cannot be decided in polynomial time by a quantum computer.

The reduction also provides a quantification, in form of a security loss, of the cost that has to be achieve the security notion, relative to the cost that an adversary has to break the underlying intractability assumption. The specific limitation will be defined at a local level.

The review of the following cryptography components and their respective security notions are in support of the remaining manuscript. Table 2.1 summarizes the notation of cryptographic algorithms, the details of which are in the subsequent paragraphs. We denote a particular scheme used in subscript, for example, $\textsc{Encrypt}_{\text{ABC}}$ if the algorithm corresponds to the encryption scheme ABC. We omit the label if the used algorithm is clear from the context.

Remark 1 (Notions and reductions in cryptography, abridged and extended from [Sch20])**.** *In complexity theory "problem P reduces to problem Q", written as $P \leq Q$, formally means that:*

$$\begin{aligned} &\exists \textit{ algorithm for } Q \Rightarrow \exists \textit{ algorithm for } P \\ &\nexists \textit{ algorithm for } P \Rightarrow \nexists \textit{ algorithm for } Q\,, \end{aligned} \tag{2.1}$$

thus showing that if no algorithm to solve problem P exists, than also no algorithm to solve problem Q exists.

In cryptography, the security of a protocol Π is expressed by a security notion X, written as X^{Π}, relative to an assumption or problem A. For a reduction *the converse of the setting in complexity theory is expressed: "security of X^{Π} reduces to the hardness of problem A", written as $X^{\Pi} \leq_{cr} A$, means that*

$$\begin{aligned} &\exists \textit{ algorithm for } X^{\Pi} \Rightarrow \exists \textit{ algorithm for } A \\ &\nexists \textit{ algorithm for } A \Rightarrow \nexists \textit{ algorithm for } X^{\Pi}\,. \end{aligned}$$

In the setting of cryptography "algorithm for" refers to an algorithm or adversary that breaks a security notion. As such, a protocol remains secure, if the problem A remains difficult.

When comparing two security notions, X, Y, cryptographers say "X is stronger than Y", corresponding to the same implication as Equation (2.1) where $P = X$ and $Q = Y$, where the security notion X usually has a more powerful *adversarial model than Y, and is therefore more difficult to achieve. Equivalently, we often write "A is a stronger assumption/problem than B", corresponding to the same implication as Equation (2.1) where $P = A$ and $Q = B$, meaning that A is the* easier *problem to solve.*

KEY ENCAPSULATION MECHANISM. A KEM as in Definition 2.2.1 is a cryptographic protocol used to securely transmit a (symmetric) key using asymmetric cryptography. Some KEMs, especially those in the NIST post-quantum competition, have a non-zero chance that the decapsulation $\textsc{Decaps}(sk, c)$ on input of the ciphertext and the secret key fails, returning an error symbol $\perp$. This is denoted as *decryption failure*, where c is a *failing ciphertext*.

Definition 2.2.1 (Key Encapsulation Mechanism)**.** *A KEM kem is a triplet $(\textsc{KeyGen}, \textsc{Encaps}, \textsc{Decaps})$ of Probabilistic Polynomial Time (PPT) algorithms: $\textsc{KeyGen}$ generates a secret key $sk \in \mathcal{SK}$ and a public key $pk \in \mathcal{PK}$, $(pk, sk) \leftarrow \textsc{KeyGen}(1^\lambda)$, $\textsc{Encaps}$ generates a key $k \in \mathcal{K}$ and a ciphertext $c \in \mathcal{C}$ as $(k, c) \leftarrow \textsc{Encaps}(pk)$, and $\textsc{Decaps}$ decapsulates the ciphertext to a shared secret, $\{k', \perp\} \leftarrow \textsc{Decaps}(sk, c)$, returning either a key or an error symbol $\perp$. The KEM is δ_{kem} correct, if k' is equal to k with probability at least $1 - \delta_{kem}$.*

A common security notion for key encapsulation mechanisms is Indistinguishability under Chosen Plaintext Attack (IND-CPA) as in Definition 2.2.2. In the game-based security a challenger samples a bit b^* and a key pair (pk^*, sk^*), creates an encapsulation ciphertext c^*, and passes pk^*, c^* and k^* to the adversary, where k^* is either a key from an encapsulation, or a random key, depending on b^*.

Definition 2.2.2 (IND-CPA KEM)**.**

$$Adv^{\text{IND-CPA}}_{kem}(A) := \left| \mathbb{P}\left[b{=}b' : \begin{array}{c} (pk, sk) \leftarrow \textsc{KeyGen}(1^\lambda); b \xleftarrow{\$} \{0,1\} \\ (k_0, c) \leftarrow \textsc{Encaps} pk; k_1 \xleftarrow{\$} \mathcal{K} \\ b' \leftarrow A(pk, c, k_b) \end{array} \right] - \frac{1}{2} \right|$$

A KEM is said to be IND-CPA secure if the running time of the adversary is polynomial in the security parameter λ, and the probability of the adversary winning the game is negligible. i. e.,

$$Adv^{\text{IND-CPA}}_{kem}(A, \lambda) \in negl\,.$$

A stronger security notion of KEMs is the IND-CCA, where the adversary additionally gets access to a decapsulation oracle. The advantage of an adversary winning an IND-CCA security game can be expressed using the following Definition 2.2.3.

Definition 2.2.3 (IND-CCA KEM, abridged from [BHK15] [BHK09, Fig. 3].)**.**

$$Adv_{KEM}^{\text{IND}-\text{CCA}}(A) := \left| \mathbb{P}\left[b{=}b' : \begin{array}{c} (pk, sk) \leftarrow \textsc{KeyGen}(1^\lambda); b \xleftarrow{\$} \{0,1\} \\ (k_0, c) \leftarrow \textsc{Encaps}pk; k_1 \xleftarrow{\$} \mathcal{K} \\ b' \leftarrow A^{\textsc{Decaps}}(pk, c, k_b) \end{array} \right] - \frac{1}{2} \right|$$

A KEM is said to be IND-CCA secure if running time of the adversary is polynomial in the security parameter and their probability to win the game is negligible:

$$Adv_{kem}^{\text{IND-CCA}}(A, \lambda) \in negl\,.$$

Remark 2. *We extend Remark 1 with an example: IND-CCA security is stronger than IND-CPA security, since the model gives the adversary an additional decapsulation oracle. Similarly for hardness assumptions, consider the Decisional Diffie–Hellman (DDH)*[1] *and Computational Diffie–Hellman (CDH)*[2] *assumption. The DDH is stronger than the CDH assumption, since an algorithm for CDH gives an algorithm to solve DDH.*

Combining security notions and assumptions, we point out a few implications: Consider the security notions X, Y with X stronger than Y and the assumptions A, B with A stronger than B. Let Π be some protocol.

Assume, $X^\Pi \leq_{cr} A$, then one can make a statement about Y, since per definition non-existence of an algorithm for to solve X implies non-existence of an algorithm to solve Y. However, we cannot make a statement about B.

$$(X^\Pi \leq_{cr} A) \Rightarrow (Y^\Pi \leq_{cr} A)$$
$$(X^\Pi \leq_{cr} A) \not\Rightarrow (X^\Pi \leq_{cr} B)$$

Similarly, assume, $Y^\Pi \leq_{cr} B$, which implies reduction to the hardness of A, since there exists an algorithm for B, there also exists an algorithm for A, but not a reduction to B.

$$(Y^\Pi \leq_{cr} B) \Rightarrow (Y^\Pi \leq_{cr} A)$$
$$(Y^\Pi \leq_{cr} B) \not\Rightarrow (X^\Pi \leq_{cr} B)$$

[1]The DDH assumption is as follows: Let $\langle g \rangle = \mathbb{G}$. Given g, g^a, g^b, H, decide if $H = g^{ab}$ or if H is uniformly random in $\mathbb{G}$.

[2]The CDH assumption is as follows: Let $\langle g \rangle = \mathbb{G}$. Given g, g^a, g^b, compute g^{ab}.

Signature Scheme. A signature as in Definition 2.2.4 is a cryptography protocol used to provide authenticating using asymmetric cryptography.

Definition 2.2.4 (Signature Scheme)**.** *A signature scheme sig is a triplet $(\textsc{KeyGen}, \textsc{Sign}, \textsc{Verify})$ of PPT algorithms: $(vk, sk) \leftarrow \textsc{KeyGen}(1^\lambda)$ samples a verification or public key $vk \in \mathcal{VK}$ and a signature key $sk \in \mathcal{SK}$. $\sigma \leftarrow \textsc{Sign}(sk, M)$ uses sk to generate a signature $\sigma \in \mathcal{S}$ for some message $M \in \mathcal{M}$. $b \leftarrow \textsc{Verify}(vk, M, \sigma)$ uses vk to check whether σ is a valid signature for the message M and outputs the result as a bit $b \in \{0, 1\}$.*

A common security notion for signature scheme is the Existential Unforgeability under Chosen Message Attack (EUF-CMA). The advantage of an adversary winning an EUF-CMA security game can be expressed using the following Definition 2.2.5.

Definition 2.2.5 (EUF-CMA, , abridged from [BS23, Attack Game. 13.1])**.**

$$Adv_{sig}^{\text{EUF-CMA}}(A) := \left| \mathbb{P}\left[1 \leftarrow \textsc{Verify}(vk, M', \sigma') : \begin{array}{c} (vk, sk) \leftarrow \textsc{KeyGen}(1^\lambda) \\ M', \sigma' \leftarrow A^{\textsc{Sign}}(sk) \\ \text{M' not queried to } \textsc{Sign} \end{array} \right] \right|$$

A signature is said to be EUF-CMA secure if the advantage of the adversary A is negligible:

$$Adv_{sig}^{\text{EUF-CMA}}(A, \lambda) \in negl\,.$$

A weaker definition is that of Universal Unforgability (UUF), where the adversary has to provide a valid signature on a message chosen by the challenger that was not signed by the challenger before as in Definition 2.2.6.

Definition 2.2.6 (UUF)**.**

$$Adv_{sig}^{UUF}(A) := \left| \mathbb{P}\left[1 \leftarrow \textsc{Verify}(vk, M^*, \sigma^*) : \begin{array}{c} (vk, sk) \leftarrow \textsc{KeyGen}(1^\lambda) \\ M^* \leftarrow \{0,1\}^* \\ \sigma^* \leftarrow A^{\textsc{Sign}}(vk, M^*) \\ M^* \text{ not queried to } \textsc{Sign} \end{array} \right] \right|$$

A signature is said to be UUF secure if the advantage of the adversary A is negligible in the security parameter λ:

$$Adv_{sig}^{\text{UUF}}(A, \lambda) \in negl\,.$$

Random Oracle and Ideal Cipher Model. The random oracle model is a powerful tool for security reductions, that allows to give entities access to a truly random function as an oracle. In practice, random oracles are instantiated with calls to cryptographic hash functions, which are not truly random functions, but for which it is computationally difficult to find preimages, second preimages and collisions.

Random oracles are theoretical tools that provide many desirable properties, such as programmability, i. e., where a reduction can choose the range of the random function "on-the-fly" during the reduction and dependent on an adversaries query. A formal treatment of the random oracle model can be found in [BS23, Sec. 8.10.2]. Whenever this manuscript talks of a random-oracle, the reader can think of a truly random function, unless otherwise specified.

[BS23] Boneh and Shoup, *A Graduate Course in Applied Cryptography*

Similarly, an Ideal Cipher (IC) is the corresponding tool to model an idealized block-ciphers, describing the map $\{0,1\}^* \times \{0,1\}^\lambda \to \{0,1\}^*$. Briefly speaking, an ideal cipher models a block cipher as a random permutation, which is fully defined by a uniformly random key. The ideal cipher and random oracle model have been shown to be equivalent [HKT10].

[HKT10] Holenstein, Künzler, and Tessaro, "Equivalence of the Random Oracle Model and the Ideal Cipher Model, Revisited"

Pseudo-Random Function Pseudo-random functions are defined over a security game, where the adversary can query an oracle with an input x, and the challenger responds with the evaluation of the pseudo-random function under a uniformly random key and an input, as in Definition 2.2.7.

Definition 2.2.7 (Pseudo-Random Function)**.** *Let PRF* : $\mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ *be an efficiently computable function and* $\mathcal{K}$ *be a key space. Define an oracle that returns either PRF*$(k^*, x) \in \mathcal{Y}$ *for a uniformly random but fixed string* k^**, or they return RF*(x) *for a uniformly random but fixed function RF* $\in \mathcal{Y}^{\mathcal{X}}$*. The choice of output is depending on a uniformly random but fixed challenge bit.*

PRF is called a Pseudo Random Function (PRF) up to $\lambda_{PRF} \in negl$*, if no polynomially bounded adversary can efficiently distinguish the two cases with advantage larger than* λ_{PRF}*.*

$$Adv_{PRF}(A,\lambda) := \Big|\mathbb{P}\left[1 \leftarrow A^{PRF(k^*,\cdot)}(1^\lambda)\middle| k^* \xleftarrow{\$} \mathcal{K}\right] - \mathbb{P}\left[1 \leftarrow A^{RF(\cdot)}(1^\lambda)\middle| RF \xleftarrow{\$} \mathcal{Y}^{\mathcal{X}}\right]\Big| \leq \lambda_{PRF}\,.$$

ONE-WAY FUNCTIONS Similarly to Pseudo-Random Functions, one-way functions are defined as in Definition 2.2.8.

Definition 2.2.8 (One-Way Function)**.** *An efficiently computable function $owf : \mathcal{X} \to \mathcal{Y}$ is called One-Way Function (OWF), if for every polynomially bounded adversary and for every $\lambda \in \mathbb{N}$, the adversary cannot find a preimage to a given, random image except with advantage $\varepsilon_{owf} \in negl(\lambda)$:*

$$Adv^{owf}_{owf}(A,\lambda) := \mathbb{P}\left[owf(x^*) = owf(x') \mid x' \leftarrow A(1^\lambda, owf(x^*)), x^* \leftarrow \mathcal{X}\right] \leq \varepsilon_{owf}$$

A keyed hash function is closely related to a key-less hash function, except that the mapping is fully defined by an additional input, the key, as in Definition 2.2.9.

Definition 2.2.9 (Keyed Hash Function, abridged from [BS23, Def. 8.2])**.** *An efficient, deterministic computable function $\mathcal{H}_K : \{0,1\}^\lambda \times \{0,1\}^m \to \{0,1\}^n$ is called a keyed hash function, if it maps from a key space and a large message space into a small digest space, i. e., $m > n$.*

The second-preimage resistance for keyed hash functions as in Definition 2.2.10 is defined similar to the second-preimage resistance of key-less hash functions, where the hash function family in question is fully determined by a key k. The key is chosen uniformly at random by a challenger. The adversary get oracle access to the keyed hash function in question along with a preimage M; their advantage is defined over their ability to provide a second-preimage M' that maps to the same image $\mathcal{H}_k(M)$.

Definition 2.2.10 (2nd Preimage Resistance, abridged from [HRS16], [BS23, Def. 8.6])**.**

$$Adv^{\mathsf{SPR}}_{\mathcal{H}_K}(A) := \left|\mathbb{P}\left[M \neq M' \wedge \mathcal{H}_K(M) = \mathcal{H}(M') : \begin{array}{r} M \xleftarrow{\$} \{0,1\}^m \\ K \xleftarrow{\$} \{0,1\}^\lambda \\ M' \leftarrow A^{\mathcal{H}_k}(M) \end{array}\right]\right|$$

Definition 2.2.10 is sometimes considered as single-function, single-target second-preimage resistance. If there is more than one target M or one more than one function $\mathcal{H}$ this is explicitly mentioned, and thus we omit the single-function and single-target prefix in the rest of the manuscript.

Part II

COSTING ADVERSARIES ON POST-QUANTUM CRYPTOGRAPHY

Summary

Quantum computing appears to be, in 2024, one of the most significant threats of the 21st-century to cryptography, with the potential to break widely used encryption methods. While symmetric cryptography appears to be largely *quantum-resistant*, i. e., it requires *only* to double the length of the security parameter, some asymmetric cryptographic protocols are much more vulnerable to large, scalable quantum computers. Quantum computing does not only introduce novel attacks using quantum algorithms, but also turns attacks that were considered infeasible into potential threats due to a quantum speedup. For cryptography to be secure for the coming decades we may need to turn to new intractability assumption, based on computational problems believed to be hard to solve even for quantum computers – post-quantum cryptography has emerged.

The most prominent post-quantum cryptosystems are the candidates of the NIST competition [Nat17]. Figure 2.1 gives an overview over the candidates of each round, along with the mathematical domain they are associated with. The first round of the competition started in 2017 with 69 candidates. In the subsequent years the schemes, their implementations and computational assumptions have been challenged to rule out insecurities. After the second and the third round, seven protocols were left, of which one KEM and three signature schemes were chosen as *finalists* to be standardized in 2022. The remaining three candidates moved on to a forth round as *alternative* finalists – one of which was broken in the same year.

[Nat17] National Institute for Standards and Technology, *Post-Quantum Cryptography Call for Proposals*

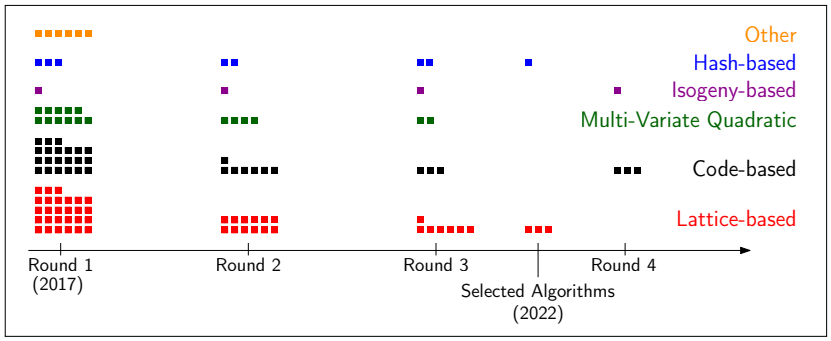


Figure 2.1: Overview of the candidates in the NIST post-quantum competition along with the mathematical domain they are based in. Note that "Other" includes Mersenne-based cryptosystems.

While losing a finalist might seem unfortunate, this is precisely the purpose of publicly reviewing the code and the specifications: to identify attacks, flaws, implementation bugs, side-channel vulnerabilities, and other security issues, leaving only the most secure candidates standing. The review process does not guarantee the absence of flaws, however, it increases the confidence in the underlying concepts and intractability assumptions. In order to quantify the security of cryptographic scheme and decide what accounts for an attack, one must carefully define the model of security, and evaluate the practicability of the most well-known algorithm. Particularly, the following questions need to be addressed:

What is a suitable cost model for (quantum) attacks?

When is a (quantum) attack considered practical?

To facilitate the discussion, the NIST gives a framework defining five security levels, each of which is equivalent to breaking an instance of either the Advanced Encryption Standard (AES) or the Secure Hash Algorithm (SHA) with a quantum-circuit of restricted depth. This is motivated by the belief, that both, AES and SHA, are relatively resistant against quantum attacks, i. e., doubling the key length restores the desired security against quantum attackers. The cost of breaking either AES or SHA with a quantum computer is then the de-facto standard to consider a cryptographic scheme secure. That means, an attack against a protocol claiming to achieve security level X is considered valid, if the *expected* quantum cost to attack this scheme is below the cost of breaking a instantiation of AES which achieves security level X. While the framework allows to provide concrete numbers to compare the cost of an attack, it does not define what constitutes *quantum cost* or *classical cost*. For example, it is not clear, whether applying a single quantum gate has the same cost as applying a single classical gate. At the time of writing, in 2024, the quantum technology is not mature enough to answer this question definitely.

In order to quantify the cost of classical and quantum algorithms we review appropriate cost models in Chapter 3 and explore the NIST security level framework in more detail. Subsequently we summarize the technical details of various candidates of the NIST post-quantum competition in Chapter 4, and review the underlying intractability assumptions as well as most-well known algorithms to approach these. The remainder of the part is then dedicated to report on our contribution to the public analysis of the NIST post-quantum competition, the results of which we summarize in the following paragraphs.

Contribution 1. In Chapter 5, which is based on [TD20a], we analyze the security of Mersenne number based cryptography, which was initially introduced [Agg+17a] as single bit encryption scheme. In 2017, Aggarwal et al. [Agg+17b] and Szepieniec [Sze17] refined the scheme and submitted KEMs based on the *Mersenne Low Hamming Combination* problem to the NIST post-quantum competition.

The Mersenne-prime cryptosystem Ramstake was proven to achieve IND-CCA security under this intractability assumption, particularly, the submission and implementation claimed 128-bits of quantum security. Both of the Mersenne-based submissions feature decryption failures, i. e., the decryption algorithm has a low, but non-zero probability of failure.

We introduce a novel attack on Mersenne-based cryptosystems, showing that IND-CCA security does not hold in the event of decryption failures. We show that the information leaked from such failing ciphertexts can be used to gain information about the secret key. As such, we present an attack exploiting this information to break the IND-CCA security of Ramstake. Our attack takes as input a set of decryption failures, i. e., ciphertexts that do not decrypt correctly, and outputs a probability distribution for the bits of the secret key. We demonstrate how the output can be used to apply an attack

[TD20a] Tiepelt and D'Anvers, "Exploiting Decryption Failures in Mersenne Number Cryptosystems"

[Agg+17a] Aggarwal et al., *A New Public-Key Cryptosystem via Mersenne Numbers*

[Agg+17b] Aggarwal et al., *Mersenne-756839*

[Sze17] Szepieniec, *Ramstake*

[Beu+19] Beunardeau et al., "On the Hardness of the Mersenne Low Hamming Ratio Assumption"

due to Beunardeau et al. [Beu+19] at significantly reduced complexity to recover the full secret.

We implemented our attack on a simplified version of the code of Ramstake submitted to the NIST competition, and made it freely available, along with instructions to reproduce all of our figures. In Chapter 5 we show how to extract the secrets from an interaction with the protocol using decryption failures. Our attack demonstrates that a *good* estimate of the secret can be extracted from 2^{12} decryption failures, which corresponds to querying 2^{74} ciphertexts in the original scheme. Using the information from these ciphertexts, the exact secrets can be extracted in about 2^{46} quantum computational steps.

Contribution 2. In Chapter 6, we analyze the security of the hash-based signature scheme SPHINCS$^+$ [Hül+20], which was a candidate in the second and third round of the NIST competition when we analyzed the scheme, and has been selected as a finalist in 2022. The results of this chapter have been published as [BT21a].

[Hül+20] Hülsing et al., *SPHINCS+-Submission to the 3rd round of the NIST post-quantum project*

[BT21a] Berger and Tiepelt, "On Forging SPHINCS$^+$-Haraka Signatures on a Fault-Tolerant Quantum Computer"

The SPHINCS$^+$ signature scheme is constructed from WOTS, FORS and XMSS scheme, all of which are combined in a large hypertree. The security of SPHINCS$^+$ is based on the difficulty to find a second preimage of the underlying hash functions, which can be instantiated with either SHAKE-256, SHA-256 or Haraka. We explore various possibilities to attack the individual signature components and compare how each component performs against a quantum preimage attack. To that end, we identify the position in the hypertree most suited for a universal forgery from a second preimage in the SPHINCS$^+$ scheme.

Subsequently, we evaluate the cost of implementing Quantum Amplitude Amplification (QAA) based search for finding a preimage, and quantify the resources required to mount such an attack on a fault tolerant quantum computer. As such, we present an attack that performs better, to the best of our knowledge, than previously published attacks (in 2021). Our analysis focuses on the Haraka hash function [Köl+16], since this has not been analyzed previously, but also provide estimates for SHAKE-256 as a comparison. As such, the analysis is limited to SPHINCS$^+$-128.

[Köl+16] Kölbl et al., "Haraka v2 - Efficient Short-Input Hashing for Post-Quantum Applications"

We propose that the weakest link is the XMSS authentication path for a given WOTS$^+$ public key, as this allows a universal forgery attack. Our most promising attack on SPHINCS$^+$-128-Haraka requires about $1.6 \cdot 2^{86}$ logical quantum gates. The same circuit to attack SPHINCS$^+$-128-SHAKE-256 has about logical $1.2 \cdot 2^{86}$ gates. In the setting of fault-tolerant quantum computing, our attack requires $1.55 \cdot 2^{101}$ *logical-qubit-cycles* and uses $1.94 \cdot 2^{20}$ physical qubits, where a *logical-qubit-cycle* is considered the quantum equivalent cost of a single classical hash-function invocation. A logical qubit cycles is computed as the product of number of logical qubits and surface code cycles required to implement the quantum circuit. In contrast, the previously most well-known generic attack on the hash function requires about 2^{129} classical hash function invocations. Performing our attack with the SHAKE-256 hash function instead requires $1.17 \cdot 2^{99}$ logical-qubit-cycles on $1.03 \cdot 2^{23}$ physical qubits. The number can be reproduces with our open implementation.

CONTRIBUTION 3. In the last chapter of this part, Chapter 7, we explore *meaningful* lower bounds on the cost of attacking lattice based cryptosystems with quantum lattice enumeration. Out of the 69 initial candidates of the NIST competition, 27 were based on lattice cryptography. Three of the finalists, the KEM Kyber [Sch+22] and the signature schemes Dilithium [Lyu+22] and Falcon [Pre+22], are based on lattice assumption.

As part of our contribution [Bin+24] we propose a new classical-quantum algorithm to perform lattice enumeration under a depth constraint, i. e., when limiting the maximum number of consecutive gates in a quantum circuit. This limitation is part of the framework proposed by NIST (cf. Section 3.4). To estimate the resources required to run our algorithm and to understand the practical applicability of quantum enumeration, we propose various *meaningful* upper and lower bounds on individual procedures of the algorithm.

We apply our results to estimate the cost of attacking the Kyber KEM. Our results for combined classical-quantum enumeration suggest that current quantum enumeration with cylinder pruning techniques are unlikely to provide practical speedups against cryptographic instances of lattice problems in a MAXDEPTH setting. Specifically, we identify that unless the distribution of the number of nodes in lattice enumeration trees is subject to a significant Jensen's gap (cf. Definition 7.1.1), Kyber-1024 is likely unaffected by quantum backtracking, even at MAXDEPTH $= 2^{96}$ and even for the most generous-to-the-adversary model. The cases for Kyber-512 and Kyber-768 are slightly more nuanced. Indeed, in our model analysis we adopt strict enough lower bounds that quantum speedups appear potentially plausible. However, we emphasize that this does not result in "breaks" or contradictions to Kyber's claims: First, our analysis is excessively generous when underestimating predictable overheads, and second, running classical-quantum enumeration in these costs would require 2^{64} of Quantum Accessible Random Access Memory (QRACM), which is controversial in the literature. We provide a tool that allows to estimate the cost of quantum lattice enumeration along with instructions to reproduce all of our figures, tables and plots.

[Sch+22] Schwabe et al., *CRYSTALS-KYBER*

[Lyu+22] Lyubashevsky et al., *CRYSTALS-DILITHIUM*

[Pre+22] Prest et al., *FALCON*

[Bin+24] Bindel et al., "Quantum Lattice Enumeration in Limited Depth"

3

Quantum Algorithms and Cost Models

This chapter reviews quantum algorithms used throughout the first part as attacks on cryptographic protocols, and subsequently metrics to quantify the cost of classical and quantum algorithms.

Parts of this chapter have been taken verbatim from our publications, i. e., [TD20a; TD20b; BT21a; BT21b; Bin+24; Bin+23].

3.1 Quantum Computing

We introduce the notation necessary to follow the manuscript, and by no means provide a complete introduction to quantum computing – for that we recommend the book of Nielsen and Chuang [NC11], as well as the lecture notes of Aaronson [Aar23] and De Wolf [De 23], which we also used as a point of reference.

[NC11] Nielsen and Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition*

[Aar23] Aaronson, *Introduction to Quantum Information Science Lecture Notes*

[De 23] De Wolf, *Quantum Computing: Lecture Notes*

Quantum States. Quantum states $|\phi\rangle$ are rays in Hilbert space $\mathcal{H}$, a complex vector space with an inner product. A k dimensional Hilbert space $\mathcal{H}^k$ corresponds to a 2^k dimensional complex vector space $\mathbb{C}^{2^k}$, associated with a basis. Commonly, the *computational basis*[1] is used, which corresponds to the set of vectors with a single one in the i^{th} entry of the vector.

We write quantum states as state vectors in *Dirac*-notation, were the *ket*-notation $|\cdot\rangle$ denotes a normalized, unit vector, and the *bra*-notation $\langle\cdot|$ denotes its conjugate transpose. We commonly use lower-case greek letters for quantum states. Quantum states evolve via unitary operations which we denote U, i. e., the unitary implementing function f is denoted U_f, where $U_f^\dagger$ is its conjugate transpose.

[1] For a single qubit the computational basis is $\left\{ |0\rangle = \begin{pmatrix}1\\0\end{pmatrix}, |1\rangle = \begin{pmatrix}0\\1\end{pmatrix} \right\}$.

Quantum Circuit Model. A quantum circuit is a directed acyclic graph that represent the evolution of quantum states from a fixed initial, to a final state. Such a circuit consists of quantum and classical wires, as well as gates corresponding to unitary evolution or measurement. The initial state is on the left side, the final state is on the right. Quantum circuits are laid out in time, rather than space, such that the order of gates matters. Figure 3.1 shows an example for a generic quantum circuit. When talking about quantum circuits we sometimes refer to *registers* rather than states, writing lower case letters $|a\rangle$ for individual qubits and upper case letters $|A\rangle$ for a multi qubit register.

Wires of quantum circuits never cross, or split up. This is due to the *No Cloning* Theorem [NC11, Box 12.1], which states that no quantum state can be copied. Indeed, there is no unitary operation that performs the map $U|\psi\rangle \otimes |0\rangle \mapsto U|\psi\rangle \otimes |\psi\rangle$. The splitting wire in Figure 3.1 is a *controlled* operation: This means, the controlled gate is applied, if and only if the controlling qubit is set to one. A gate can be controlled by one or multiple qubits.

Quantum Memory and Uncomputation. The evolution of a quantum state, and thus the quantum gates, is described by a unitary matrix, Accordingly,

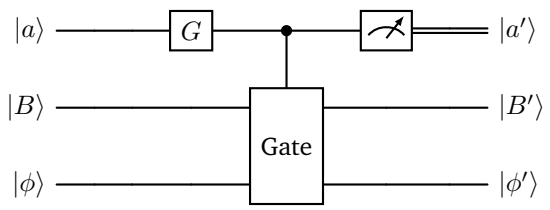


FIGURE 3.1: Example of a quantum circuit with initial state $|b\rangle \otimes |B\rangle \otimes |\phi\rangle$, where register $|B\rangle$ consists of n qubits. G is a single qubit gate, *Gate* a controlled multi qubit gate. After the measurement, the double wire represents to the collapsed quantum state.

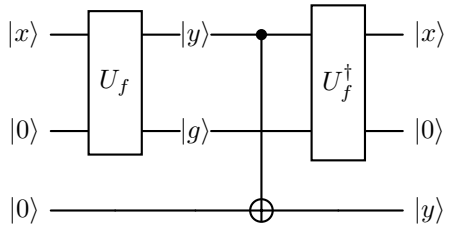


FIGURE 3.2: Copy-uncompute circuit that allows to reset ancillary quantum registers to a known state.

every quantum circuit is reversible – particularly, only reversible operations can be implemented in quantum circuits. While this seems to be a drawback at first, Bennett [Ben89] showed, that, informally, every non-reversible Turing machine running in time T and using space S can be simulated by a reversible Turing machine in time $O(T^{1+\epsilon}), \epsilon > 0$ and space $O(S \log T)$. As such, any classical algorithm can be translated to a quantum circuit with at most polynomial overhead in both time and space.

[Ben89] Bennett, "Time/Space Trade-Offs for Reversible Computation"

Along with the results on reversible computing, Bennett [Ben89] also introduced the concept of "copy-uncompute", which allows to reset a register of ancillary qubits to a known state. Consider a unitary U_f that acts on Hilbert space $\mathcal{H}_1 \otimes \mathcal{H}_2$, where $\mathcal{H}_2$ are intermediate results that are not further used in the computation. After evolving the joined quantum state with U_f, the states in the Hilbert spaces can be entangled. Then, we cannot "reset" (for example, by measuring) the quantum state corresponding to $\mathcal{H}_2$ without disturbing the other states. Indeed we would have to preserve this quantum state until the end of the whole computation.

Instead, one can apply the copy-uncompute trick depicted in Figure 3.2: First, the circuit is extended with an additional register corresponding to a vector in $\mathcal{H}_3$ that will hold the final result. Then the initial state is $|x\rangle |0\rangle |0\rangle \in \mathcal{H}_1 \otimes \mathcal{H}_2 \otimes \mathcal{H}_3$. After applying U_f, the state is $|y\rangle |g\rangle |0\rangle$, where $|g\rangle$ is some unknown *garbage* state as intermediate output from computing the unitary. Next the computation result $|x\rangle$ is "copied"[2] to the additional register, resulting in state $|y\rangle |g\rangle |y\rangle$. In a final step, the unitary U_f in uncomputed by applying the conjugate transpose $U_f^\dagger$, which resets the garbage to a known state, resulting in $|x\rangle |0\rangle |y\rangle$. Now the quantum register $|0\rangle \in \mathcal{H}_2$ is in a known state that can be re-used in further computation. Note that we often have to keep both $|x\rangle$ and $|y\rangle$ (the in- and the output) of the computation until measurement to preserve reversibility. Bennett [Ben89]

[2]Note that this does not correspond to an actual copy of the state. The name stems from the fact that same content is now in both registers, but with the caveat that they are entangled.

used the copy-uncompute trick to show their bound on the additional memory required to construct a reversible circuit. In a nutshell, the additional memory comes from constructing a binary tree of height $\log T$, that computes the intermediate result of every computational step, and later uncomputes this.

QRACM. We consider the use of QRACM. This can be thought of as a classical array $(a_1, \ldots, a_n)$ that can be read by a quantum computer into a state $\sum_{i \leq n} |a_i\rangle$ in $O(n \text{ poly}(\log(n)))$ operations [GLM08; Kup11; JR23]. All our algorithms use a polynomial amount of qubits, and some of our approaches require an exponential-size QRACM. The exact amount of qubits required will be given for each algorithm individually.

[GLM08] Giovannetti, Lloyd, and Maccone, "Quantum Random Access Memory"

[Kup11] Kuperberg, *Another subexponential-time quantum algorithm for the dihedral hidden subgroup problem*

[JR23] Jaques and Rattew, *QRAM: A Survey and Critique*

3.1.1 *Quantum Amplitude Amplification*

In this manuscript we use quantum amplitude amplification in the form of Grover's search algorithm, as depicted in Figure 3.3, and as quantum walk as described in Section 3.1.2. We use Grover's algorithm in Chapter 5 to estimate the asymptotic cost of recovering the secret key of Mersenne-number based cryptography, and in Chapter 6 to estimate the cost of computing a second preimage of a hashfunction. We use quantum walks to estimate the cost of quantum lattice enumeration in Chapter 7.

Let D be a set (or "database") and $T \subset D$ be a subset of targets. Let $f\colon D \to \{0,1\}$ be a membership oracle, that identifies membership of elements in T, that is, $f(x) = 1$ if and only if $x \in T$. Let U_D be a unitary that maps $|0\rangle \mapsto \sum_{i \in D} |i\rangle$, U_f be such that $U_f |x\rangle = (-1)^{f(x)} |x\rangle$, and $U_0 := I_n - 2 |0^{\otimes n}\rangle \langle 0^{\otimes n}|$ be a unitary that inverts the phase of $|0\rangle$, where n corresponds to the number of qubits required to represent the elements in the database. Define the operator $Q := (-U_D U_0 U_D^{-1}) U_f$. QAA allows to find an element in T by applying operator Q times to a state initially holding $|D\rangle$. At the end of the computation the final state is $\alpha |T\rangle + \beta |E\rangle$, where $|E\rangle$ corresponds to some error state that does not overlap with the target state $|T\rangle$. A circuit corresponding to Grover's algorithm is depicted in Figure 3.3.

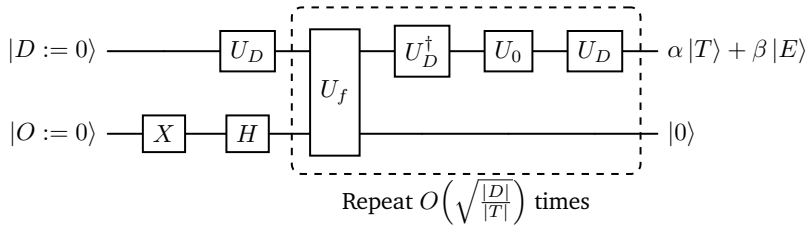


FIGURE 3.3: Generic quantum circuit for Grover's algorithm where U_D generates a superposition over a database, U_f marks a *target* state and $U_0 = I_n - 2 |0^{\otimes n}\rangle \langle 0^{\otimes n}|$ is the inversion about the mean of the amplitudes. The sequence of gates comprising U_D and the inversion of the mean are referred to as Diffusion. At the end of the computation the amplitude of the target states α is close to one. The states $|E\rangle$ with amplitude β are error states.

The success probability, i. e.probability $|\alpha|^2$ to measure $x \in T$, when measuring the final state in Figure 3.3 relies on performing the correct

number of iterations. Let the initial distribution of the quantum state be

$$\sum_{i \in D} |i\rangle = \frac{1}{\sqrt{t}} |\psi_G\rangle + \frac{1}{\sqrt{|D| - t}} |\psi_B\rangle \ ,$$

where $|\psi_G\rangle$ corresponds to the target states $x \in T$ and $|\psi_B\rangle$ to all other states $x \in D \setminus T$. After j iterations, the state is described by Lemma 1.

Lemma 1. *(State of QAA, abridged from [Boy+05, Sec 3]) Let $\theta = \frac{|T|}{|D|}$ the initial probability to measure a target element. After j iterations of quantum amplitude amplification, it holds that the state is $Q^j |\psi\rangle = \frac{1}{\sqrt{|T|}} \sin((2j + 1)\theta) |\psi_G\rangle + \frac{1}{\sqrt{|D|-|T|}} \cos((2j + 1)\theta) |\psi_B\rangle$.*

As such, the quantum state encodes an element in T with large probability after $\frac{\pi}{4}\sqrt{\frac{|D|}{|T|}}$ applications of operator Q. Even if $|T|$ is not known in advance, it can be shown[Bra+02] that an element of T can be extracted with $O(\sqrt{|D|/|T|})$ applications of Q. Finally, if the amplitude of the error state $|E\rangle$ is to large, one can perform probability amplification by combining multiple executions of Grover's algorithm. We note that we address this individually for the specific use cases if required.

[Bra+02] Brassard et al., *Quantum amplitude amplification and estimation*

3.1.2 *Quantum Backtracking from Quantum Walks*

BACKTRACKING. Backtracking can be implemented as a depth-first search (DFS) algorithm that allows exhaustively finding "marked" leaves on trees, where a marked leaf corresponds to an unknown assignment to a set of variables. Let P be a predicate that takes as input a variable assignment and outputs *one*, if the assignment evaluates to true, INDETERMINATE if the solution is ambiguous and *zero* otherwise.

A backtracking algorithm starts in an initial configuration of the variables and evaluates the predicate P on a *new* assignment to the variables repeatedly, If the predicate returns INDETERMINATE, it assign a value to the *next* variable resulting in a new assignment, and computes the predicate again. This is repeated until all possible values of all variables have been checked, or until the predicate returned true on an assignment. This procedure generates a backtracking tree as in Definition 3.1.1.

Definition 3.1.1 (Backtracking Tree)**.** *Given a predicate P and a heuristic to decide which variable is assigned next, a backtracking tree $\mathcal{T}$ is an undirected graph representing variable assignments in a depth-first-search. Each node in the tree represents a partial assignment to a subset of variables ordered by the heuristic, such that the nodes at level i instantiate the variables $x_1, ..., x_i$. The children of a node at level i are all assignments to the variable x_{i+1} such that $P(x_1, ..., x_i, x_{i+1}) \neq 0$.*

QUANTUM PHASE ESTIMATION. Given a unitary operator U with eigenvector $|u\rangle$ and eigenvalue $e^{2\pi i\varphi}$, $\varphi \in [0, 1)$, Quantum Phase Estimation (QPE) finds the value of φ by repeatedly applying the operator U. Quantum phase estimation can be used to implement a quantum walk as detailed in the following paragraphs.

Theorem 1 (Quantum Phase Estimation, abridged from [Cle+98, Sec. 5])**.** *There exists a quantum circuit that finds φ to a precision of s bits with a probability of $1-\epsilon$ using $s'+|u|$ qubits, where $\epsilon = s + \lceil \log(2+\frac{1}{2\epsilon})\rceil$. The algorithm uses 2^ϵ controlled-U operations and $O\left((\epsilon)^2\right)$ other gates.*

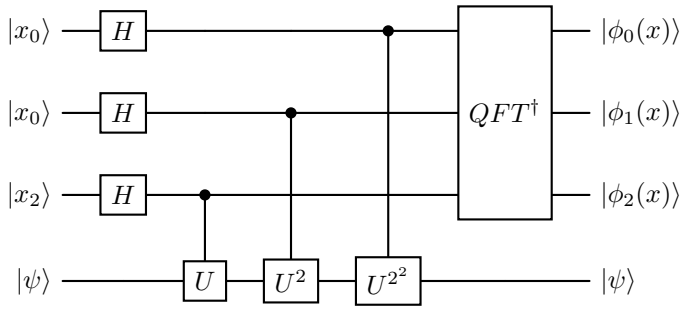


FIGURE 3.4: Quantum circuit for quantum phase estimation of operator U with precision $s = 3$. QFT† is the conjugate transpose of the quantum Fourier transform.

QUANTUM BACKTRACKING. In [Mon18], Montanaro introduced quantum backtracking algorithms using a quantum walk. A quantum walk is the quantum equivalent of a random walk on an undirected graph $G = (V, E)$ where a subset of the nodes is *marked*, if they are a solution to the search problem. [Sze04] showed that, starting from an initial state in a stationary distribution, a marked vertex can be found in time $O(\sqrt{H})$ of the classical hitting time H. [Bel13, Sec 2.1] extended the result to arbitrary starting distributions allowing to walk graphs that are defined locally, i. e., given an oracle that computes a neighbor on input of a node. [Bel13] modeled the graph as an electrical network, where each edge is assigned a weight and each node a resistance. Then [Bel13, Thm 4] shows that a quantum walk can detect a marked node with failure probability at most $\frac{2}{3}$ after applying QPE with precision $\frac{1}{\beta\sqrt{RW}}$, for some constant $\beta > 0$ and where R is the sum of the resistance on the path from the initial distribution to a marked node and W is the total sum of all weight of the graph. The precision of the QPE is inverse to the leading cost of the quantum walk.

[Mon18] Montanaro, "Quantum-Walk Speedup of Backtracking Algorithms"

[Sze04] Szegedy, "Quantum Speed-Up of Markov Chain Based Algorithms"

[Bel13] Belovs, *Quantum Walks and Electric Networks*

Montanaro provided algorithms for detecting [Mon18, Alg. 2] (cf. Algorithm 3.5) and finding [Mon18, Sec. 2.3] (cf. Algorithm 3.6) marked vertices in a tree, both achieving an asymptotic speedup over classical backtracking. Both of these algorithms are based on a common quantum walk, QPE($\mathcal{W}$), where $\mathcal{W}$ is the operator that corresponds to a single step of the quantum walk using the predicate P_R to decide if a node is marked.

The quantum walk applies the operator $\mathcal{W}$ consecutively WQ($\mathcal{T}, \mathcal{W}$) often to a state corresponding to a superposition of the nodes in the backtracking tree $\mathcal{T}$. By the proof of [Mon18, Lem. 2.4], the eigenvectors of $\mathcal{W}$ are the states that admit a path from the root to a marked node. Measuring the root node after a sufficient number of steps allows to identify whether a path to a marked node exists with false positive probability $\leq 1/4$ and false negative probability $\leq 1/2$.

The detection algorithm DETECTMV [Mon18, Alg. 2] consists of repeating QD($\mathcal{T}$) $\coloneqq \lceil \epsilon \log(1/\delta_{\text{DMV}})\rceil$ times QPE, for some constant $\epsilon > 0$, to output

"marked node exists" or "no marked node exists" with a failure probability of at most δ_{DMV}. We determine bounds on values for ϵ and δ_{DMV} in Section 7.1.1 (with a more detailed analysis in [Bin+23, App. H]) and refer to the resulting number of calls to the QPE within DETECTMV as $\text{QD}(\mathcal{T})$.

[Bin+23] Bindel et al., *Quantum Lattice Enumeration in Limited Depth*

DETECTMV $(R_A, R_B, \delta_{\text{DMV}}, h, \beta, \epsilon, x)$

countAccept $= 0$
repeat $\text{QD}(\mathcal{T}) := \lceil \epsilon \log(1/\delta_{\text{DMV}}) \rceil$ times:
  if $\text{QPE}_x\left(R_B, R_A, \text{prec} := \beta/\sqrt{\#\mathcal{T}h}\right) = 1$ **then**
    countAccept += 1
if countAccept $\geq \frac{3 \cdot \text{QD}(\mathcal{T})}{8}$
    return "marked node exists"
else
    return "no marked node exists"

FIGURE 3.5: Algorithm to detect a marked vertex where the QPE_x is executed on the tree rooted at node x with precision *prec*.

FINDMV $(R_A, R_B, \delta_{\text{DMV}}, h, \beta, \epsilon, x)$

if x is marked
  return x and terminate.
if DETECTMV $(R_A, R_B, \delta_{\text{DMV}}, h, \beta, \epsilon, x)$ outputs "marked vertex exists"
  for $c \in \text{children}(x)$
    FINDMV $(R_A, R_B, \delta_{\text{DMV}}, h-1, \beta, \epsilon, c)$
return "no marked vertex exists".

FIGURE 3.6: Algorithm to find a marked vertex in a tree rooted at node x of height h. The function $\text{children}(x)$ returns all the children of node x.

Theorem 2 (Quantum Backtracking, abridged from [Mon18, Theorem 1.2])**.** *Let $\mathcal{T}$ be a backtracking tree of size at most $\#\mathcal{T}^u$ with degree $O(1)$. Let $P(x)$ be a predicate that returns true if and only if x is a marked node. For any $0 < \delta_{DMV} < 1$, DETECTMV outputs "marked node exists" if there exists $x \in \mathcal{T}$ such that $P(x) =$ true and "marked node does not exist" otherwise, with failure probability at most δ_{DMV}. The algorithm performs $O\left(\sqrt{\#\mathcal{T}^u n} \log(1/\delta_{DMV})\right)$ evaluations of P, using poly(n) qubits.*

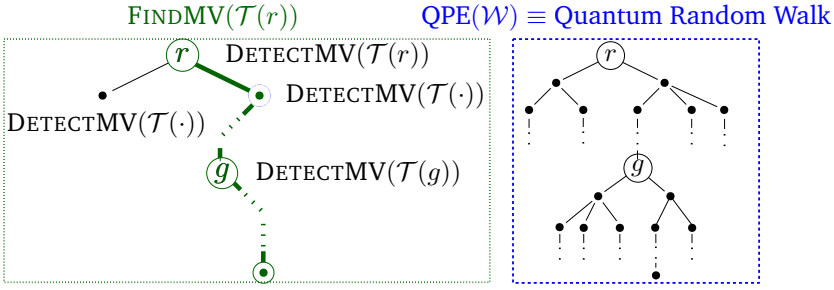


FIGURE 3.7: Simplified overview of the procedures FINDMV, DETECTMV and the tree searched by the first call to the quantum phase estimation.

For the algorithm returning a marked node in $\mathcal{T}$, Montanaro suggests to perform classical depth-first-search on $\mathcal{T}$ by using DETECTMV as a predicate. Figure 3.7 shows how the two procedures and the quantum walk act on a

simplified backtracking tree. DETECTMV would be called on the children c_i of the root node, until one on the subtrees rooted at c_i, for some i, returns "marked node exists". It would then proceed to search for a child of c_i spawning a subtree with a marked node, and so on, until reaching a marked leaf. For a tree of height h where each node has at most $\mathcal{C}$ children, FINDMV will call DETECTMV $\mathrm{DF}(\mathcal{T}) \leq 2 \cdot h \log \mathcal{C}$ times, in the worst case.

If no upper bound on the number of nodes of $\mathcal{T}$ is known, the search can be repeated with growing values of $\#\mathcal{T}^u = 2^0, 2^1, 2^2, \ldots$, resulting in an additional runtime factor of $O(\log \#\mathcal{T})$. Montanaro also shows that overestimating the tree-size does not affect the quantum walk's success probability.

QUANTUM BACKTRACKING WITH TREE SIZE ESTIMATION. It is important to note that the runtime of Montanaro's quantum walk [Mon18] scales with the upper bound on the tree size. Ambainis and Kokainis [AK17] introduce an algorithm that uses the quantum walk of [Mon18] as a subroutine, and the runtime of which is bounded (asymptotically) by the number of nodes explored by the corresponding classical backtracking algorithm. They achieve this by searching trees with growing upper bound on the tree size, until either a marked node has been found, or until the runtime of Montanaro's algorithm is reached. In the latter case, [AK17] concludes that no marked vertex is in the tree. Briefly speaking, the algorithm of [AK17] repeats a DFS for growing values of $i = 0, 1, \ldots \log(\#\mathcal{T})$, or until a marked node is found. The DFS applies a quantum tree size estimation (cf. [AK17, Alg. 1]) to the subtree rooted at every node visited, until the estimated size is less than 2^i. They then apply the quantum walk [Mon18] to that subtree with 2^i as an upper bound on the tree size.

[Mon18] Montanaro, "Quantum-Walk Speedup of Backtracking Algorithms"

[AK17] Ambainis and Kokainis, "Quantum algorithm for tree size estimation, with applications to backtracking and 2-player games"

3.2 CLASSICAL COST MODEL

In the context of the NIST's post-quantum competition classical protocols are subject to an analysis under the assumption that an adversary has access to a quantum computer. The protocols under investigation are *low* level protocols for either key encapsulation, or for digital signatures. Correspondingly, the primary cost metric is computational cost. The communication cost, defined as the number of interactions that the adversary has with any party, is not considered in this analysis. These limitations are implicit and depend on the computational power of the adversary. For instance, a polynomially bounded adversary can have a polynomial number of interactions, but they are not part of the security assessment. These resources play a more significant role in higher level protocols (cf. Chapters 9 and 10 of Part III).

The classical cost metric is the computational cost, which be can most readily defined as the cost for a single computational step. This computational step can refer to taking a single step on a processor or a Turing machine, but could also mean to compute some efficient function. Another resource might be the memory required to run an algorithm. If the memory cost is polynomial in the security parameter, the precise cost is often disregarded. In this manuscript, we use the conventional asymptotic or Landau notation as outlined in Section 2.1.

TRANSFORMING CLASSICAL TO QUANTUM ALGORITHMS. The relationship between classical and quantum cost is well-established in the asymptotic limit. In 1989 Bennett [Ben89, Thm. 1] demonstrated that any classical, deterministic Turing machine running in time T and space S can be simulated by a reversible Turing machine using time $O(T^{1+\epsilon}),\ \epsilon > 0$ and space $O(S \cdot \log T)$. Reversible Turing machines are equivalent to unitary operations, where the Turing machine can be interpreted as an operator U and its inverse as the conjugate transpose $U^\dagger$. Consequently, the theorem of [Ben89] provides a bound on the cost and memory required when translating classical to quantum computations.

[Ben89] Bennett, "Time/Space Trade-Offs for Reversible Computation"

In practice, translating classical procedures into quantum circuits proves to be a significant challenge, with an increase in the cost in the range of polynomial factors. Reasons for this include, but are not limited too, inefficient arithmetic, dependence on memory access, inability to parallelize, etc. The algorithm with the lowest asymptotic complexity may not always be the one that is the most efficient to translate into a quantum setting, and thus may not result in the most practical attack. Conversely, it may also happen that a function can be computed significantly more efficient on a quantum computer, such that relevant instances have small constants, i. e., close to 0. As a result there are relevant real-world problems with asymptotically large cost that can be solved rapidly. Consequently, evaluating costs in an asymptotic model can result in an over- or underestimation of the cost of an algorithm.

The following section is devoted to quantifying the cost of quantum computation on a more practical level. All the below models can be considered in the asymptotic setting, as is done initially in all three Chapters 5 to 7. Additionally, we will also make assumptions on concrete implementation for quantum circuits that lead to concrete values representing the cost of an adversary relative to the *current* technological state. Based on these assumption we provide a concrete cost estimation of our attacks (cf. Chapters 6 and 7).

3.3 QUANTUM COST MODEL

The layered architecture for quantum computing [Jon+12] is a systematic framework that allows to analyze the resources to execute quantum algorithms relative to a concrete hardware implementation. The architecture is divided into five layers as shown in Figure 3.8. The two *lowest* layers deal with the hardware implementation of the quantum computer, i. e., how to read and write data onto a physical device, and how quantum gates are applied on those devices. The layers provide as an interface a basic set of faulty gates and qubits. These two layers are strongly dependent on the concrete physical device. In this manuscript, we only consider the three top-most layers, that allow to quantify resources relative to quantum gates that appear in the *quantum circuit model*.

[Jon+12] Jones et al., "Layered Architecture for Quantum Computing"

The third layer, the fault-tolerant layer, quantifies the necessity for error correction of the faulty gates and qubits provided by the previous layers. The layer "outputs" reliable qubits and gates, where reliable is defined as the probability of causing a fault that is below a given threshold. In order to

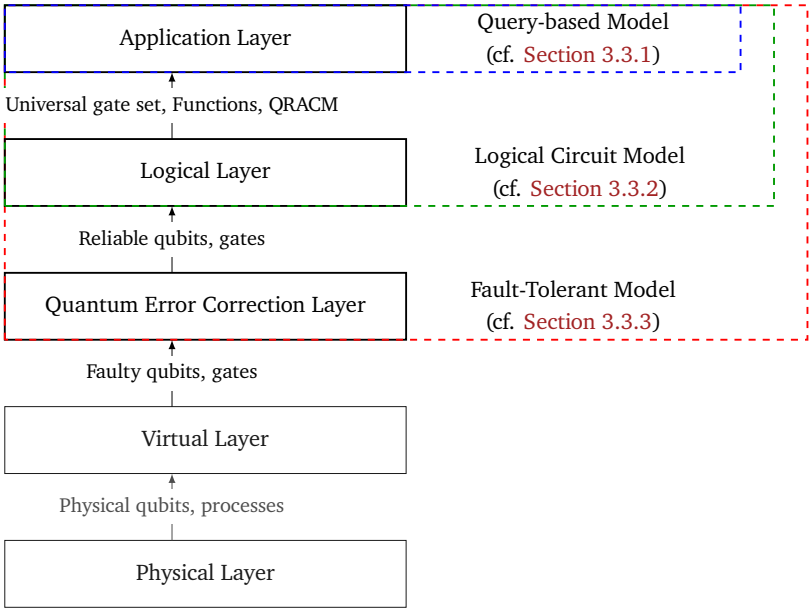


FIGURE 3.8: The layered quantum computing architecture of [Jon+12] ranges from the physical implementation to the application layer. The top three layers are of particular interest in resource estimation. The application layer and often considers attacks relative to the number of queries to some oracle. The logical layer considers the cost of individual quantum logic gates. The quantum error correction layer considers the cost to translate faulty quantum resources, into reliable logical resources, allowing to make assumptions on adversarial costs relative to a specific quantum computing architecture. The remaining two layers are much dependent on the physical implementation of the actual qubits and gates.

provide fault-tolerant resources, the layer deploys error correction and other techniques that combine multiple faulty resources into fewer fault-tolerant resources. While this requires assumptions on the cost of implementing these techniques, it also provides the most detailed and realistic resource estimation.

The logical layer expands the gates and qubits to provide a universal set of quantum gates and considers the cost of these individual quantum gates, allowing a better comparison of classical and quantum resources. The number of universal, logical gates is closely related to the classical cost of a computation. The application layer corresponds to the interface to the "user" and requires the fewest assumptions about technology. It which quantifies the cost of algorithms relative to computational steps, or to invocations of functions and oracles.

3.3.1 *Application Layer: Query-based Model*

The application layer is the interface visible to a "user" who is executing a quantum algorithm. The resource analysis conducted at this layer is independent of the underlying implementation and is solely concerned with the *efficient* computational steps required to execute an algorithm. The interpretation of a *computational step* can vary, from individual gates that are applied, to calling any function or procedure that can be computed efficiently. This setting implies a conservative bound on the cost of the adversary, given that the concrete implementation of the function is disregarded. The model lower bounds this at unit cost.

In the remaining manuscript we sometimes refer to this layer as the query-based model, assuming black-box access to an oracle that computes a function on any input on-demand. We use this model to estimate the cost of our decryption-query attack on Mersenne-based cryptosystems in Chapter 5, to identify the *best* oracle for Grover's algorithm in Chapter 6, and when exploring generous-to-the attacker lower bounds for quantum lattice enumeration in Chapter 7.

3.3.2 *Logical Circuit Layer: Circuit-based Model*

Another, more precise, metric is the circuit based model, in which a black-box is instantiated with an algorithm or quantum circuit. At a low level, we can decompose any algorithm into a (universal) set of quantum gates. The function of the logical circuit level is to provide these quantum gates. The cost of implementing the quantum circuit can then be quantified by counting the number of logical resources in the form of quantum gates and qubits. In some cases it may be desirable to count a different resource, such as arithmetic operations. We use this model to estimate the cost of implementing hash functions for our analysis in Chapter 6, and considering concrete cost estimates for quantum lattice enumeration in Chapter 7.

DECOMPOSITION. We rely on the de facto standard for universal quantum gates in the literature [Alb+19b; Jaq+20; Jon+10], the Clifford+T gates. This choice also allow us to benefit from a large literature on quantum circuits as well as tools for resource estimates such as Q# [Mic20].

On this layer, we assume that the gates and qubits provided are fault-tolerant and that non-local operations are cheap, i. e., without additional assumptions on decoherence, error correction or time steps for implementing an operator on a physical device. These are the same assumptions made by tools such as Q# that we use in part of our work to estimate some upper bounds on circuit costs in this model.

[Alb+19b] Albrecht et al., *Estimating quantum speedups for lattice sieves*

[Jaq+20] Jaques et al., "Implementing Grover Oracles for Quantum Key Search on AES and LowMC"

[Jon+10] Jones et al., "Layered Architecture for Quantum Computing"

[Mic20] Microsoft, *Q# Language Specification*

NUMBER OF GATES. The number of gates is commonly referred to as GCOST (or G-cost) in the literature. The use of the GCOST is motivated by the observation that, in practice, quantum gates are not a physical device, but an operation performed on the quantum state. Such operation is likely managed by a classical micro-controller, meaning that the G-cost is a lower bound on the cost of evaluating gates of a quantum circuit. As such cost also consumes classical resources, it can be compared to the cost of classical algorithms [JS19, Def. 2.4]. Figure 3.9 shows a simplified example of the GCOST in a quantum circuit.

[JS19] Jaques and Schanck, "Quantum Cryptanalysis in the RAM Model: Claw-Finding Attacks on SIKE"

Definition 3.3.1 (G-cost, abridged from [JS19, Def 2.4])**.** *A quantum circuit that uses G logical Clifford+T gates has a GCost of $\theta(G)$ RAM operations.*

CIRCUIT DEPTH. It is also necessary to choose a cost metric to optimize the circuit. In this manuscript, we follow the approach of the NIST [Nat17, Sec. 4.A.5], who suggest that quantum circuits may be limited to a fixed running time, i. e., a fixed quantum circuit depth.

[Nat17] National Institute for Standards and Technology, *Post-Quantum Cryptography Call for Proposals*

The circuit depth can be defined as the longest path from the input state to the output state, if the circuit is considered as a directed graph with

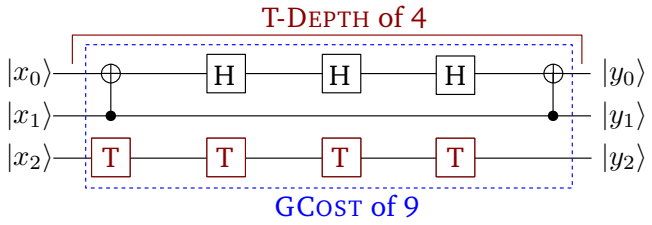


FIGURE 3.9: Simplified example for GCOST and T-DEPTH in a quantum circuit from the Clifford+T gate set.

quantum gates as nodes. That means, we assume all independent parts are computed in parallel. This is a standard assumption (for example [Jaq+20; Bai+23; Alb+20b]), as most error correction schemes are active, meaning it is costly to maintain a memory, whether or not a computation is done with it. As such, circuit depth can be seen as an analogous measure to the runtime of a classical computation, by considering that applying a gate must take a non-zero amount of time, and is therefore often used to express the asymptotic cost of quantum algorithms. We consider that the depth of the circuit is a good representation of the time needed to run the algorithm.

Further, we make a crucial assumption regarding circuit depth: We exclusively measure *T-depth*, that is the circuit depth when only taking into consideration T gates, as preparation of these is expected to be the most time-consuming part of practical quantum computation [Jon+10; Fow+12]. Figure 3.9 shows a simplified example of the T-DEPTH in a quantum circuit.

[Jaq+20] Jaques et al., "Implementing Grover Oracles for Quantum Key Search on AES and LowMC"

[Bai+23] Bai et al., "Concrete Analysis of Quantum Lattice Enumeration"

[Alb+20b] Albrecht et al., "Estimating Quantum Speedups for Lattice Sieves"

[Jon+10] Jones et al., "Layered Architecture for Quantum Computing"

[Fow+12] Fowler et al., "Surface codes: Towards practical large-scale quantum computation"

3.3.3 *Quantum Error Correction Layer: Fault-Tolerant Model*

We could try to be even more precise, aligning with the architectural approach of existing quantum computer prototypes, and take into account the quantum error-correcting or a limited qubit connectivity. The model for fault-tolerant quantum computing represents such a model.

From a cost perspective it is the most precise approach, as its smallest unit is the physical implementation of a gate or a qubit. Conversely, it also carries the burden of relying on our current understanding of how quantum computers may be constructed. In particular, the quantum computing model used in this manuscript considers the quantum computing architecture of [Jon+12; Fow+12] first presented in 2012.

In summary, the model receives a number of virtual, noisy qubits and gates, and outputs fault-tolerant qubits and logical gates. To achieve this, the architecture deploys error-correcting *surface codes* [Fow+12] and magic state distillation Figure 3.10 to improve the quality of both the gates and the qubit states. Surface codes and magic state distillation are used as black-boxes, that means, we do not consider the exact inner working, and only provide parameters to instantiate the black-box. The interaction of error-correcting codes and magic state distillation in quantum computing architecture layers is shown in Figure 3.10.

As with the logical cost analysis in Section 3.3.2, the T-gates appear to be the most expensive resource [Jon+12; Fow+12][Bin+23, Sec. 2.3]. Therefore, our primary metric of concern is the cost of providing fault-tolerant

[Jon+12] Jones et al., "Layered Architecture for Quantum Computing"

[Bin+23] Bindel et al., *Quantum Lattice Enumeration in Limited Depth*

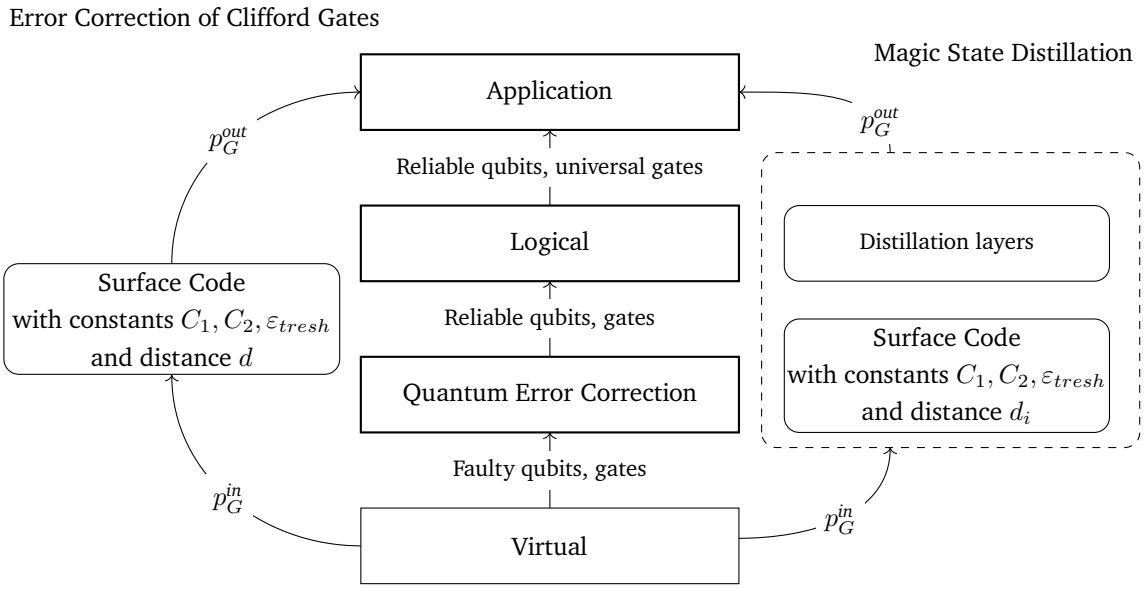


FIGURE 3.10: Fault-tolerant architecture for quantum computing. The layers gets as input a set of faulty qubits and gates, associated with a failure probability p_G^{in}. The output are the gates and qubits with failure probability at most p_G^{out}. The thresholds are achieved by deploying magic state distillation and surface codes that provide error correction based on a set of constants, and the variable surface code distance d_1, d_2.

T-gates, which are processed using a process called *magic state distillation*, which provides an error-corrected T-gate.

To estimate the required quantum error correction, one must make assumptions on the physical implementation, i. e., the error probability of the quantum computer provided by the virtual layer.

To instantiate these probabilities, we rely on Assumption 3.3.1, which approximates the state of the art of quantum computing at the time of writing [Amy+16; FDJ13]. We use these for comparability, but note that other values have also been suggested, for example in [Jon+12].

[Amy+16] Amy et al., "Estimating the Cost of Generic Quantum Pre-image Attacks on SHA-2 and SHA-3"

[FDJ13] Fowler, Devitt, and Jones, "Surface code implementation of block code state distillation"

[Jon+12] Jones et al., "Layered Architecture for Quantum Computing"

Assumption 3.3.1. *There exists a quantum computer architecture that provides physical gates with error probability at most* $p_G^{in} \approx 10^{-5}$ *[Amy+16; Roe+17; Jaq19].*

We make an additional assumption (cf. Assumption 3.3.2) which simplifies the cost estimation.

Assumption 3.3.2. *The quantum gates of a circuit are distributed uniformly across all layers.*

While Assumption 3.3.2 does not hold per se, it does so for Grover's algorithm (cf. Section 3.1.1) over multiple Grover iterations, since each iteration performs the same computations.

FAULT-TOLERANT COST ESTIMATION. On a high level, the following steps summarize the estimation of the fault-tolerant cost of an algorithm:

1. Determine the number of logical quantum gates to construct a quantum circuit for the algorithm.

2. Determine a bound on the failure probability of each gate, such that the application of all gates in the circuit succeeds with a probability of about one. This is referred to as the *output* success probability p_G^{out} of each gate.

3. Determine the resources required to implement surface codes and magic state distillation to transform gates with error probability p_G^{in} to gates with error probability at most p_G^{out}.

In the following we detail the resources and assumption of the third step.

SURFACE CODES. Surface codes are a family of quantum error correcting codes that correct errors introduced into a quantum state during the application of a gate. In this manuscript we use these codes as a black-box: The code is defined over constants C_1, C_2 and ε_{thresh} that are determined by the respective implementation. The error correcting code turns non-T gates with error probability p_Q^{in} into gates with error probability $p^{out} \approx C_1(C_2 p_Q^{in})/\varepsilon_{thresh}^{\lfloor (d+1)/2 \rfloor}$ [Jon+12, Sec. IV.B]. Furthermore, each surface code is associated with a surface code distance d, which is determined by the input and output error probabilities. We follow Assumption 3.3.3 to determine the parameters in our cost estimation.

[Jon+12] Jones et al., "Layered Architecture for Quantum Computing"

Assumption 3.3.3. *There exists a surface code with constants $C_1 = 1$ and $C_2/\varepsilon_{thresh} = 80$, and surface code distance d that maps gates with error probability p_G^{in} to gates with error probability at most*

$$p_G^{out} \leq (80 p_G^{in})^{\lfloor d+1/2 \rfloor}, \tag{3.1}$$

Let p_G^{in} be the failure probability of a quantum gate. For each logical qubit that the gates operate on, the surface code requires $2(d+1)^2$ physical qubits to be implemented in a surface code. [Jon+12, Sec. IV.B]. A surface code with distance d uses about d surface code cycles. A surface code takes about $\tau_{sc} = 200ns$ for a single cycle. [Amy+16; Roe+17; Jaq19]

MAGIC STATE DISTILLATION. Magic state distillation is the process of generating *magic states* in one or multiple layers of magic state distilleries. A *magic state* is a special quantum state that is used to construct a single, fault-tolerant T-gate, the exact construction of which is not of interest for this manuscript i. e., magic state distillation is used as a black-box.

For each T-gate, such a magic state has to be constructed. Initially, the magic states have error probability from Assumption 3.3.1 and need to be *distilled* to have a lower error probability. This distillation is performed using layers of error correction. In this manuscript, we deploy the Reed-Muller-15-to-1 [BK05] distillation as instantiated in Definition 3.3.2. Each layer uses 15 magic states with an input error rate of $p_G^{in,i}$ and produces one magic state with lower error rate $p_G^{out,i+1} \approx 35(p_G^{in,i})^3$. We follow the work of [Amy+16] and assume that the amount of logical errors introduced during distillation is already covered in the process.

[BK05] Bravyi and Kitaev, "Universal quantum computation with ideal Clifford gates and noisy ancillas"

[Amy+16] Amy et al., "Estimating the Cost of Generic Quantum Pre-image Attacks on SHA-2 and SHA-3"

In order to reach the desired output error probability p_G^{out} one may need to compute multiple layers i of magic state distillation. Each layer uses a surface code with distance d_i.

Definition 3.3.2 (Reed-Muller Distillation [FDJ13, Sec. II]). *Let d_i be a distance of a surface code of the i^{th} layer of magic state distillation. The Reed-Muller-15-to-1 [BK05] distillation requires $10d_i$ surface code cycles for each layer and requires $16 \cdot 15^{i-1}$ logical qubits.*

[BK05] Bravyi and Kitaev, "Universal quantum computation with ideal Clifford gates and noisy ancillas"

3.4 NIST SECURITY FRAMEWORK

As part of their call for proposals, NIST [Nat17] defined five security *categories* corresponding to the hardness of breaking AES and SHA. When considering quantum algorithms, they expressed this hardness in terms of G-cost, assuming a MAXDEPTH limit, i. e., a limit to the depth of a quantum circuit available to an attacker. The five security levels along with the limitation on quantum circuit depth according to MAXDEPTH are presented in Table 3.1,

[Nat17] National Institute for Standards and Technology, *Post-Quantum Cryptography Call for Proposals*

TABLE 3.1: Security categories as suggested by NIST [Nat17, page 18].

Security Category	As difficult to break as	quantum gates	classical gates
I	AES128	2^{170}/MAXDEPTH	2^{143}
II	SHA256	—	2^{146}
III	AES192	2^{233}/MAXDEPTH	2^{207}
IV	SHA384	—	2^{210}
V	AES256	2^{274}/MAXDEPTH	2^{272}
—	SHA512	—	2^{274}

We investigate the cost of quantum algorithms when imposing the limit MAXDEPTH to the maximum depth a circuit can achieve while maintaining state coherence. This consideration follows from observing that currently state decoherence seems to be one of the main hurdles to achieving large-scale quantum computation. As part of the call for proposals for its post-quantum cryptography standardization process, NIST [Nat17] proposed the three possible values of 2^{40}, 2^{64} or 2^{96} for MAXDEPTH, corresponding to the "[...] approximate number of gates that presently envisioned quantum computing architectures are expected to serially perform in [...]"[Nat17, Sec. 5.A.4] a year, a decade or in a millennium, respectively.

This limitation means that care should be paid to any circuit parallelization required to stay within MAXDEPTH circuit depth when measuring the cost of long-running quantum algorithms as these do not always trivially parallelize, such as in the case of Grover's search [Zal99]. Accordingly, any quantum computation requiring a deeper circuit would run multiple circuits in parallel or sequence, if possible.

[Zal99] Zalka, "Grover's quantum searching algorithm is optimal"

We consider that the depth of the circuit is a good representation of the time needed to run the algorithm. As a result, we assume all gates that are executed on independent states are computed in parallel as is common in similar estimations [GE21; Jaq+20]. Respectively, the number of quantum gates to break AES is estimated by NIST to be 2^X/MAXDEPTH, for example $2^{170}/2^{40} = 2^{130}$ for AES-128 as given in Table 3.1. A more nuanced analysis of the cost of breaking AES was given by [Jaq+20], resulting in the numbers of Table 3.2, which are the basis for evaluating the security of KEMs.

[GE21] Gidney and Ekerå, "How to factor 2048 bit RSA integers in 8 hours using 20 million noisy qubits"

[Jaq+20] Jaques et al., "Implementing Grover Oracles for Quantum Key Search on AES and LowMC"

TABLE 3.2: Analysis of GCOST to break AES under MAXDEPTH [Jaq+20, Table 10, 12].

MAXDEPTH	AES Variant	GCOST
2^{40}	AES-128	$1.07 \cdot 2^{117}$
2^{40}	AES-192	$1.09 \cdot 2^{181}$
2^{40}	AES-256	$1.39 \cdot 2^{245}$
2^{64}	AES-128	$1.07 \cdot 2^{93}$
2^{64}	AES-192	$1.09 \cdot 2^{157}$
2^{64}	AES-256	$1.39 \cdot 2^{221}$
2^{96}	AES-128	$1.34 \cdot 2^{83}$
2^{96}	AES-192	$1.09 \cdot 2^{126}$
2^{96}	AES-256	$1.39 \cdot 2^{190}$
∞	AES-128	$1.34 \cdot 2^{82}$
∞	AES-192	$1.50 \cdot 2^{115}$
∞	AES-256	$1.38 \cdot 2^{148}$

LIMITS OF THE NIST METRICS. The NIST metrics were designed to give a security goal and allow comparisons between candidates. As all metrics, part of them is arbitrary, and they were not designed to accurately define what could be computable in the long term. In particular, the classical and quantum security levels are incomparable, and a break of a system due to a quantum attack does not imply the most efficient way to attack it will be, in the long run, quantum. Still, post-quantum cryptography has to do some optimistic assumptions on the capabilities of quantum computers, as otherwise pre-quantum schemes would suffice.

4

Post-Quantum Cryptography

4.1 MERSENNE NUMBER BASED CRYPTOGRAPHY

In 2017, [Agg+17a] introduced Mersenne number-based cryptography, which was based on the *new* "Low Hamming Combination" intractability assumption. The cryptosystems get their name from the modulus, a Mersenne number, which is defined as a number of the form $p = 2^n - 1$, with n an integer. The NIST post-quantum competition featured two Mersenne-based KEMs, Ramstake [Sze17] and Mersenne-756839 [Agg+17b], the security of which was based on a variant of this new assumption. In Chapter 5 we present an attack on these submissions.

A notable property of arithmetic modulo a Mersenne number is that the Hamming weight (HW), defined as the number of ones in the binary representation, does not increase during modular operations in the ring $\mathbb{Z}/p\mathbb{Z}$. Further, multiplication by a power of two in this ring is correspond to a rotational shift, i. e., performing $2^i x \mod p$ is equivalent to rotating the binary representation of x by i positions, making for efficient arithmetic operations. The binary representation of integers in the ring $\mathbb{Z}/p\mathbb{Z}$ have bit length at most n.

Parts of this chapter have been taken verbatim from [TD20a; TD20b; BT21a; BT21b; Bin+24; Bin+23].

[Agg+17a] Aggarwal et al., *A New Public-Key Cryptosystem via Mersenne Numbers*

[Sze17] Szepieniec, *Ramstake*

[Agg+17b] Aggarwal et al., *Mersenne-756839*

4.1.1 *Computational Hardness Assumptions*

In the remaining of this part, we will denote the HW of the binary representation x of an integer by $\text{HW}(x)$. Likewise, the bytewise Hamming weight of a binary string x will be written as $\text{HW}^8(x)$, which returns the number of nonzero bytes in x.

Let p be a Mersenne prime, let a, b be integers with bitwise Hamming weight ω, and G a random group element in $\mathbb{Z}/p\mathbb{Z}$. The Mersenne Low Hamming Combination (LHC) Assumption in Definition 4.1.1 states, that it is computationally difficult to distinguish $aG + b \mod p$ from a random number in $\mathbb{Z}/p\mathbb{Z}$, when given an integer and G.

Definition 4.1.1 (Mersenne Low Hamming Combination Assumption, abridged from [Agg+17b, Def. 5]). *Let $2^n - 1$ be a Mersenne prime and ω an integer such that $4\omega^2 < n \leq 16\omega^2$. The advantage of a PPT adversary to distinguish the two tuples*

$$\left(\begin{bmatrix} G_1 \\ G_2 \end{bmatrix}, \begin{bmatrix} G_1 \\ G_2 \end{bmatrix} \cdot a + \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \right) \text{ or } \left(\begin{bmatrix} G_1 \\ G_2 \end{bmatrix}, \begin{bmatrix} R_1 \\ R_2 \end{bmatrix} \right),$$

$\textsc{KeyGen}_M(1^\lambda)$

1 : $sd \xleftarrow{\$} \{0,1\}^{256}$
2 : $G \leftarrow gen_G(sd)$
3 : $a, b \leftarrow \mathcal{HW}_\omega(\mathbb{Z}_p) \times \mathcal{HW}_\omega(\mathbb{Z}_p)$
4 : $P_D \leftarrow aG + b \mod p$
5 : **return** $(pk := (sd, P_D), sk := (a, b, sd,))$

FIGURE 4.1: Key generation for Mersenne-based public-key encryption.

$\textsc{Encrypt}_M(pk := (sd, P_D), M; r)$

1 : $r_1, r_2 \leftarrow split(r)$
2 : $c, d \leftarrow \mathcal{HW}_\omega(\mathbb{Z}_p; r_1) \times \mathcal{HW}_\omega(\mathbb{Z}_p; r_2)$
3 : $G \leftarrow gen_G(sd)$
4 : $P_E \leftarrow cG + d \mod p$
5 : $S \leftarrow cP_D \mod p$
6 : $M_{ecc}, h_M = \texttt{Encode}(M)$
7 : $enc_M = M_{ecc} \oplus S\,[0 : |M_{ecc}|]$
8 : **return** $(ct := (enc_M, P_E, h_M))$

$\textsc{Decrypt}_M(ct, sk := (a, b, sd))$

1 : $G \leftarrow gen_G(sd)$
2 : $S' \leftarrow aP_E \mod p$
3 : $M'_{ecc} = enc_M \oplus S'\,[0 : |enc_M|]$
4 : $M' = \texttt{Decode}(M'_{ecc}, h_M)$
5 : **return** M'

FIGURE 4.2: Encryption and Decryption algorithm for Mersenne-based public-key encryption.

where G_1, G_2, R_1 and R_2 are chosen uniformly random in $\mathbb{Z}/p\mathbb{Z}$ and a, b_1 and b_2 are uniformly random elements in $\mathbb{Z}/p\mathbb{Z}$ with Hamming weight ω, is at most $O(2^{-\omega})$.

The best known algorithm to solve the LHC Assumption is the Slice-and-Dice attack, which we discuss in Section 4.1.3.

4.1.2 *Mersenne number Encryption Scheme*

With the LHC at hand, we define a generalized Mersenne prime encryption scheme which can be used to build an IND-CCA secure KEM using the Fujisaki-Okamoto transformation [FO99].

[FO99] Fujisaki and Okamoto, "Secure Integration of Asymmetric and Symmetric Encryption Schemes"

Choose p as a Mersenne prime and let $gen_G()$ be a pseudo-random generator that expands a seed sd into a uniformly random element in $\mathbb{Z}/p\mathbb{Z}$. Let $\mathcal{H}$ and $\mathcal{H}'$ be hash functions modeling random oracles. Let Encode and Decode be a pair of Error Correcting Code (ECC) functions that respectively encode a binary string M into an encoded binary string M_{ECC}, and that decode a noisy version of the latter string back into the original such that up to t errors can be corrected. Let $\mathcal{HW}_\omega(\mathbb{Z}_p)$ be a function that samples a uniformly random string in $\mathbb{Z}_p$ with Hamming weight exactly ω; and let $\mathcal{HW}_\omega(\mathbb{Z}_p, r)$ denote the function where the output is generated pseudorandomly from r. Let split(x) denote splitting the input into two equally sized substrings. Let further $M \in \{0,1\}^n$ be a message and $r \in \{0,1\}^\lambda$ a random string. Given these functions, the generalized Mersenne prime scheme is defined via the Algorithms 4.1 to 4.2. The public key of the encryption scheme is $pk = (sd, P_D := aG + b \mod p)$, the secret key is $sk := (sd, a, b)$

[Sze17] Szepieniec, *Ramstake*

Given these algorithms one can construct the two NIST submissions, [Sze17; Agg+17b], both of which use the Fujisaki-Okamoto transform [FO99; FO13] to achieve IND-CCA security. The transformation introduces a re-encryption step in Figure 4.2, which may cause the algorithm to either return

[Agg+17b] Aggarwal et al., *Mersenne-756839*

[FO13] Fujisaki and Okamoto, "Secure Integration of Asymmetric and Symmetric Encryption Schemes"

the message M', or a decryption failure denoted as $\perp$. This happens for [Sze17] with probability 2^{-64} and for [Agg+17b] with probability 2^{-246}.

[Sze17] Szepieniec, *Ramstake*

[Agg+17b] Aggarwal et al., *Mersenne-756839*

4.1.3 *Security*

The security of both Mersenne-based submissions, [Sze17; Agg+17b], is based on the LHC Assumption from Definition 4.1.1. In the same year of introducing Mersenne number cryptosystems (cf. Section 4.1), the Slice-and-Dice algorithm was introduced by Beunardeau et al. [Beu+19] and later refined by [Boe+18]. This algorithm is the most well-known procedure to recover the secrets of a Mersenne number cryptosystem given only the public key. In brief summary, the idea revolves around partitioning the secret key into random parts from which a lattice is constructed. If the partitioning fulfills a certain property, then the secrets correspond to short vector in the lattice that fall into the approximation factor bound [Boe+18, Lem. 5], allowing LLL to output these unique, shortest vectors but still run in polynomial time. The bottleneck is guessing a partitioning which fulfills this property, and requires a number of guesses exponential in the Hamming weight of the secrets.

[Beu+19] Beunardeau et al., "On the Hardness of the Mersenne Low Hamming Ratio Assumption"

[Boe+18] Boer et al., "Attacks on the AJPS Mersenne-Based Cryptosystem"

Attack Description. The original procedure by Beunardeau et al. was applied to a single-bit encryption scheme. We assume as input the public key $pk := (G, H = aG + b \mod p)$. Consider the binary string representation of the sparse integer a. One can partition this string into multiple parts, each representing a substring starting at bit position p_i. Interpreting the i^{th} substring $a[p_i : p_{i+1}]$ as an integer X_i gives a representation of the sparse integer as $a = \sum_i 2^{p_i} X_i$. Consider a balanced partition, i. e., all parts have *similar* bit length, $P := \{p_1, p_2, ..., p_k\}, p_i < p_{i+1}, p_i \in [0, n)$ of a (respectively $Q := \{q_1, q_2, ..., q_l\}$ of b). Then we consider the following lattice:

$$\mathcal{L}_{a,b,H} = \left\{ (X_1, ..., X_k, Y_1, ..., Y_l) \middle| \sum_{i=1}^{k} 2^{p_i} X_i G - \sum_{j=1}^{l} 2^{q_j} Y_j \equiv H \mod p \right\}. \tag{4.1}$$

The lattice defined in Equation (4.1) contains vectors representing the secrets. A basis that generates the lattice can be constructed as shown in Equation (4.2).

$$\left(\begin{array}{ccc|c} & & & 0 \\ & \mathcal{I} & & \vdots \\ & & & 0 \\ \hline 0 & \dots & 0 & 1 \\ 0 & \dots & 0 & 0 \end{array}\right. \left.\begin{array}{c} -2^{p_1} G H^{-1} \mod p \\ \vdots \\ -2^{q_l} G H^{-1} \mod p \\ \hline -H \\ p \end{array}\right) \tag{4.2}$$

Furthermore, the lattice $\mathcal{L}_{a,b,H}$ contains *malicious* vectors of the form $(0, .., 2^{p_{i+1}-p_i}, -1, 0, \hat{Y}_1, ..., \hat{Y}_k, ..., 0)$, such that $\sum_j^k \hat{Y}_j = H$. We call them malicious, because they are the shortest vectors retrieved by the LLL algorithm if care is not taken by constructing the partition. Boer et al. show that these vectors have norm about $2^{|P_i|}, 2^{|Q_i|}$ for parts P_i, Q_i. They use a heuristic due

[GN08b] Gama and Nguyen, "Predicting Lattice Reduction"

to [GN08b] to show that the malicious vectors are not the shortest vectors, if and only if the bit-length of the norm of each of the parts is larger than $n/d + \Theta(\log n)$, where $d \approx 2\omega$ is the rank of the lattice. To see this, consider a balanced partition containing ω parts, such that each part has bit length n/ω. If the norm of the malicious vector has bit-length $n/2\omega + \theta(\log n)$, the secrets have smaller norm only, if all ones in the binary expansion fall into the lower $n/2\omega$ bits of each part, such that the norm also has bit-length at most $n/2\omega$. Intuitively, sampling random partitions results in a successful attack if all ones of the secret fall into the lower half of each part.

In the following we denote a part as *correct*, if it represents a subset of the binary expansion of a secret and if all its ones are positioned in the lower half of the part. A partition is *correct*, if all parts are correct. Figure 4.3 shows an example for partitioning the binary representation of a string, such that the integer in each individual parts have either short bit-length if the ones fall into the lower half, or large bit-length otherwise.

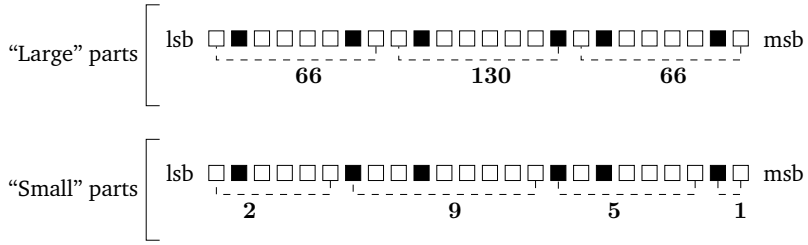


FIGURE 4.3: The figure shows a partitioning of the binary representation of an integer such that each individual part corresponds either to a large integer (top), or small integer (bottom). The black boxes represent the ones, the white boxes the zeros in the binary expansion.

Beunardeau et al. [Beu+19, Sec 2.2, Remark 1] generalized the approach to imbalanced partitions tolerating parts with larger norms and others with smaller norm. This can be achieved by scaling all parts relative to their size. Let $K_{max} = max_i\left(|R_i|\right), R_i \in \{P_1, ..., P_k, Q_1, ..., Q_l\}$ denote the bit length of the largest part. Then the scaling parameter is defined as $\kappa_{P_i} = K_{\text{max}} - |P_i|$ (respectively for Q_i). A scaled vector in the lattice is of the form $(\kappa_{P_1} X_1, ..., \kappa_{P_k} X_k, \kappa_{Q_1} Y_1, ..., \kappa_{Q_l} Y_l)$. Consider the norm of a malicious vector resulting from a part of *small* bit length. The technique ensures that its norm is scaled to exceed the norm of vector resulting from a part with *large* bit length which has is ones only in the lower half.

[Beu+19] Beunardeau et al., "On the Hardness of the Mersenne Low Hamming Ratio Assumption"

The algorithm is summarized in Figure 4.4, which finds a partitioning for the secrets a, b such that the norm of the resulting vector is small, in particular such that each one of the secret falls into the lower half of a part.

SLICEANDDICE(pk, G, p)

while True
  $P, Q \xleftarrow{\$} \mathbb{Z}_n^{2\omega}$ ▷ Sample parts.
  $B \leftarrow$ construct basis for $\mathcal{L}_{a,b,H}$ from P, Q
  $B^* \leftarrow$ LATTICEREDUCTION(B) ▷ Returns short vectors.
  if $\exists\, b_a^*, b_b^* \in B^*$ s.t. $b_a^* G + b_b^* = pk$
    return b_a^*, b_b^*

FIGURE 4.4: Slice-and-Dice algorithm to attack Mersenne-based cryptosystems.

We review a variant of the Slice-and-Dice attack in Chapter 5 to estimate the cost of recovering the secret key with additional information provided by our attack.

COST. Briefly speaking, the advantage of an adversary to find the secret key of the encryption scheme is bounded by the Hamming weight ω of the sparse integers. Finding such a partition is difficult due to the unknown structure of the secrets the instantiations of the Mersenne number cryptosystem submitted to the NIST competition. It is known to be classically bounded by $O(2^{2\omega})$, and quantumly bounded by $O(2^{\omega})$ [TS19]. This means that the schemes maintain at least 256-bits of classical and 128-bits of quantum security.

[TS19] Tiepelt and Szepieniec, "Quantum LLL with an Application to Mersenne Number Cryptosystems"

[Boe+18] gave a precise analysis and bound the fraction r of the part containing positions of ones, that would guarantee that the secret vectors are also the shortest, and that would thus allow to extract the secret using lattice reduction[1]. The exact value depends on the rank of the reduced lattice and is omitted here (cf. [Boe+18, Sec. 5.3] for details). For the sake of simplicity assume this fraction to be $r \approx n/2\omega$. Let $k = l = \omega$ be the number of parts, then the number of *correct* positions is kr/n for a and lr/n for b. For a randomly chosen partition the probability of being *correct* is about

[Boe+18] Boer et al., "Attacks on the AJPS Mersenne-Based Cryptosystem"

[1] Any SVP oracle can be used. [Beu+19] showed that the LLL algorithm is sufficient, since it performs the lattice reduction in polynomial time.

$$\left(\frac{kr}{n}\right)^{\omega} \cdot \left(\frac{lr}{n}\right)^{\omega} \approx \left(\frac{kl}{(2\omega)^2}\right)^{\omega} = \left(\frac{1}{2}\right)^{2\omega} .$$

It follows that the expected number of guesses to perform the attack is $O(2^{2\omega})$. The cost is reduced to $O(2^{1.75\omega})$ if only the indistinguishability of ciphertexts is attacked [CG20].

[CG20] Coron and Gini, "Improved cryptanalysis of the AJPS Mersenne based cryptosystem"

Enclosing the Slice-an-Dice attack into a quantum amplitude amplification (cf. Section 3.1.1), as initially suggested by [Beu+19] and later refined by [TS19], requires Grover iterations corresponding to the square root of the number of guesses, i. e., $O(2^{\omega})$.

[Beu+19] Beunardeau et al., "On the Hardness of the Mersenne Low Hamming Ratio Assumption"

Other attacks on Mersenne number cryptosystems have been proposed, however, these were targeting the implementations of the error correcting code [DAn+19b] exploiting timing variations to extract the secret keys. These attacks are not relevant for our novel attack exploiting decryption failures, which we introduce in Section 5.2.

[DAn+19b] D'Anvers et al., "Timing Attacks on Error Correcting Codes in Post-Quantum Schemes"

4.2 HASH-BASED CRYPTOGRAPHY

The concept of digital signatures, the security of which is based on the difficulty of inverting one-way functions, was first introduced by [Lam79] as a One-Time-Signature (OTS). Subsequently, these one-time signatures were developed into multi-use signature schemes utilizing a Merkle Tree, a binary tree structure where each child node is the result of applying a collision resistant function to its parent nodes. Modern implementations include the XMSS [BDH11] and the SPHINCS signature scheme [Ber+14], with a variant of the latter submitted as a candidate of the NIST post-quantum competition, and which was selected in 2022 as a finalist.

[Lam79] Lamport, "Constructing Digital Signatures from a One Way Function"

[BDH11] Buchmann, Dahmen, and Hülsing, "XMSS - A Practical Forward Secure Signature Scheme based on Minimal Security Assumptions"

[Ber+14] Bernstein et al., "SPHINCS: practical stateless hash-based signatures"

In the real world, one-way functions are realized with cryptographic hash functions. Consequently, digital signatures can be constructed from cryptographic hash functions, the security of which is based on the collision

TABLE 4.1: SPHINCS$^+$ parameters with security parameter in bits, abridged from [Hül+20, Table 3]

Parameter	Usage	Instantiation in SPHINCS$^+$-					
		128s	128f	192s	192f	256s	256f
λ	Security parameter	128	128	192	192	256	126
w	Winternitz parameter	16	16	16	16	16	16
h	Height of hypertree	63	66	63	66	64	68
d	Number of layers of hypertree	7	22	7	22	8	17
k	Number of FORS trees	14	33	17	33	22	35

resistance or the second-preimage resistance. Since cryptographic hash functions are believed to be *mostly* resistant to quantum attacks, i. e., they are subject to the quadratic quantum-speedup for preimage search, or cubic for collision search from quantum amplitude amplification (cf. Section 3.1.1), post-quantum signatures can be conjectured to be subject to this speedup only too.

4.2.1 *Computational Hardness Assumption*

The security of SPHINCS$^+$ is based on the *post-quantum distinct-function, multi-target second-preimage resistance* of a *tweakable* hash function [Hül+20, Sec. 9.1]. A tweakable hash function is a keyed hash function (cf. Definition 2.2.9), where each call to the hash function includes a key that determines the specific function used. [HRS16] shows that the security of SPHINCS$^+$ corresponds to the hardness of the (single-function, single-target) keyed second-preimage resistance of a keyed hash function (cf. Definition 2.2.10). That means, that the attack cost is independent of the number of targets.

[Hül+20] Hülsing et al., *SPHINCS+- Submission to the 3rd round of the NIST post-quantum project*

[HRS16] Hülsing, Rijneveld, and Song, "Mitigating Multi-target Attacks in Hash-Based Signatures"

4.2.2 *SPHINCS$^+$ Signature Scheme*

The SPHINCS$^+$ signature scheme[Hül+20] is a stateless, hash-based signature scheme, that was submitted as a candidate to the NIST post-quantum competition [Nat20] in 2017, and selected as a finalist in 2022. The scheme combines multiple hash-based signature schemes, namely a WOTS, FORS and XMSS scheme, which are combined into a virtual hypertree. The scheme is constructed around various parameters which determine the dimensions of the individual signature schemes and of the hypertree. For the purposes of this manuscript, the parameters from Table 4.1 are relevant. Further, throughout the scheme we will use the following variables:

[Nat20] National Institute for Standards and Technology, *Post-Quantum Cryptography Round 1*

Address $ADRS \in \{0,1\}^{256}$. The address *ADRS* is a fixed length value that ensures that the hash functions calls for each key pair and each position in the virtual tree are independent. The address is used in SPHINCS$^+$ [Hül+20, Sec. 2.7.3] to determine the hash function in WOTS$^+$ and FORS schemes, the compression function of the WOTS$^+$ and FORS public keys, as well as the compression function in the XMSS trees.

Key material The public key is vk_{SPHINCS^+}, the secret key sk_{SPHINCS^+}. Additionally, both public and secret key have a string "seed" $sd \in \{0,1\}^\lambda$ as part of the key material.

HASH FUNCTION INSTANTIATIONS. SPHINCS$^+$ deploys multiple tweakable hash functions [Hül+20, Sec. 2.7.1] of the form

$$\mathcal{H}_i : \{0,1\}^\lambda \times \{0,1\}^{256} \times \{0,1\}^{i\cdot\lambda} \to \{0,1\}^\lambda , \tag{4.3}$$

where i depends on the size of the input to be hashed.

While any (post-quantum) keyed hash function that is second-preimage resistant is suitable to be used to construct post-quantum signatures, SPHINCS$^+$ proposed to use either Haraka [Köl+16], SHA-256 or SHAKE-256 [NIS15]. The Haraka hash function is an AES based hash function with two variants, Haraka-256 and Haraka-512, mapping either from a 256 or 512 bits state block to a 256 bit output block. The SHAKE-256 hash function is an expandable hash based on the SHA-3 hash function, which in turn builds on the Keccak permutation. The exact inner working of the hash functions are not relevant for this manuscript and can be found in the respective specifications [NIS15; Köl+16]. We do not consider the SHA-256 instantiation of SPHINCS$^+$.

[Hül+20] Hülsing et al., *SPHINCS+-Submission to the 3rd round of the NIST post-quantum project*

[Köl+16] Kölbl et al., "Haraka v2 - Efficient Short-Input Hashing for Post-Quantum Applications"

[NIS15] NIST, *SHA-3 standard: Permutation-based hash and extendable-output functions*

WINTERNITZ ONE TIME SIGNATURES. A WOTS in SPHINCS$^+$ [Hül+20, Sec. 3.5] uses the hash function $\mathcal{H}_1$ to construct a hash chain that comprises the secret key sk_{WOTS^+}, the signature σ_{WOTS^+} and the public key vk_{WOTS^+}. The WOTS scheme is configured with the Winternitz parameter w and the security parameter λ, the latter of which determines the length of the bitstrings which are evaluated with the hash function $\mathcal{H}_1$. Figure 4.5 gives an example for one of such hash chains, along with the position of the secret and public key, and the signature, where $\mathcal{H}_1$ is instantiated with either SHAKE-256, SHA-256-2 or Haraka.

In the SPHINCS$^+$ scheme, the WOTS$^+$ is used to sign a string M_X and its checksum $check_{M_X}$. Let $X = M_X || check_{M_X}$ be the concatenation of the string M_X and its checksum[2]. The WOTS$^+$ signs the string X by splitting it into multiple blocks and iterating an individual hash chain for each block. The exact algorithms of key generation, signing and verification are not important. We highlight the parts important to follow this manuscript:

[2]The exact computation of the checksum is not relevant for this manuscript.

Key Generation The private key is sk_{WOTS^+} and consists of λ-bit blocks $sk_{\text{WOTS}^+,i}$, each of which corresponds to an initial node in a hash chain. The public key vk_{WOTS^+} is the hash of the collection of hashes $vk_{\text{WOTS}^+,i}$ for all blocks i at the end of a hash chain as described in the signature procedure. The public key is derived separately from the secret key.

Signature To generate a signature $\sigma^X_{\text{WOTS}^+} = (\sigma^{X_1}_{\text{WOTS}^+,1}, \sigma^{X_2}_{\text{WOTS}^+,2}, \ldots)$ the string X is divided into blocks of bitlength w. Let X_i be the i^{th} block which, interpreted in base w, corresponds to an integer between 0 and $w-1$. Then the signature of the i^{th} block is the output of $(X_i)_w$ (where $(X_i)_w$ denotes X_i in basis w) recursive evaluations of the hash function $\mathcal{H}_1$ on input $sk_{\text{WOTS}^+,i}$, i. e.,

$$\sigma^{X_i}_{\text{WOTS}^+,i} = \mathcal{H}_1(\ldots\mathcal{H}_1(vk_{\text{WOTS}^+}.sd, ADRS, sk_{\text{WOTS}^+,i}))\,,$$

where the hash function is applied $(X_i)_w$ times.

The i^{th} public key $vk_{\text{WOTS}^+,i}$ is the last node in the hash chains, after a total of w evaluations.

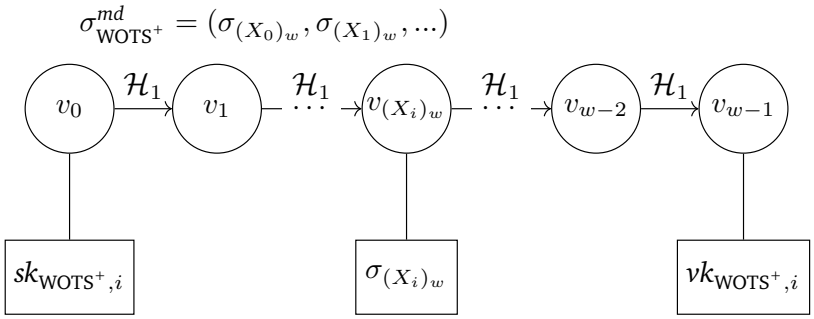


FIGURE 4.5: Structure of one hash chain of a WOTS instance, with the i^{th} secret- and public key, the i^{th} signature and the i^{th} block of the string X.

Verification To verify a WOTS$^+$ signature in the SPHINCS$^+$ scheme, the hash chain for each signature $\sigma_{(X_i)_w}$ is advanced the remaining $w - (X_i)_w$ iterations to compute the public key $vk_{\text{WOTS}^+,i}$, which is then used to compute a WOTS$^+$ public key vk_{WOTS^+}. Subsequently, the computed public key is compared to the *genuine* public key.

FOREST OF RANDOM SUBSETS. A FORS signature scheme [Hül+20, Sec. 5] is a collection, i. e., a forest, of binary hash trees that sign a message digest *md*. In case of the SPHINCS$^+$ scheme the latter is the output of a hash function. The scheme is defined by a number k of private key sets and a number t of elements in each of these sets. Each of the k sets corresponds to one binary hash tree with a root $r_i, i \in [0, k-1)$, while t is the number of leaves of each such tree. A FORS signature signs multiple blocks X_i, $i \in [k]$, of a message digest *md*. Each such block X_i is associated with the j^{th} secret key, which acts as the leaf node of the i^{th} tree. The exact mapping of blocks to leaves is not relevant for this manuscript. Figure 4.6 shows an example forest and a signature, where only the first tree is made explicit.

[Hül+20] Hülsing et al., *SPHINCS+- Submission to the 3rd round of the NIST post-quantum project*

Key Generation The secret key $sk_{\text{FORS}} = \{sk^{i,j}_{\text{FORS}}\}_{i\in[k],j\in[t]}$ is a set of $k \cdot t$ random strings. Each such random strings represents a leaf in one of the trees. The public key vk_{FORS} is the hash function output of all of the k root nodes $\{r_i\}_i$ of the trees.

Signature A signature $\sigma^{md}_{\text{FORS}}$ is a collection of k strings $sk^{i,j}_{\text{FORS}}$ (where j denotes the index of a leaf in a FORS tree) along with an authentication path $Auth_{\text{FORS},i}$ through the i^{th} binary tree, i. e., all siblings on the path from the leaf to the root node.

To sign a message digest *md*, the digest is first split into k bit strings each representing a number i between $0...t-1$, thus corresponding to the i^{th} private key. Then for each private key an authentication path in the k^{th} tree is computed, meaning that each sibling node required to reconstruct the root of the i^{th} tree is computed using function $\mathcal{H}_2$. The final signature consists of the k private key values along with the authentication paths.

Verification A signature is verified by computing the public key from the secret string and the authentication path, where the compressed public key is the hash T_k of the roots of all k trees.

$$vk^{md}_{\text{FORS}} \leftarrow \mathcal{H}(vk.sd, ADRS, r_{\text{FORS},1}, r_{\text{FORS},2}, ..., r_{\text{FORS},k}) \ .$$

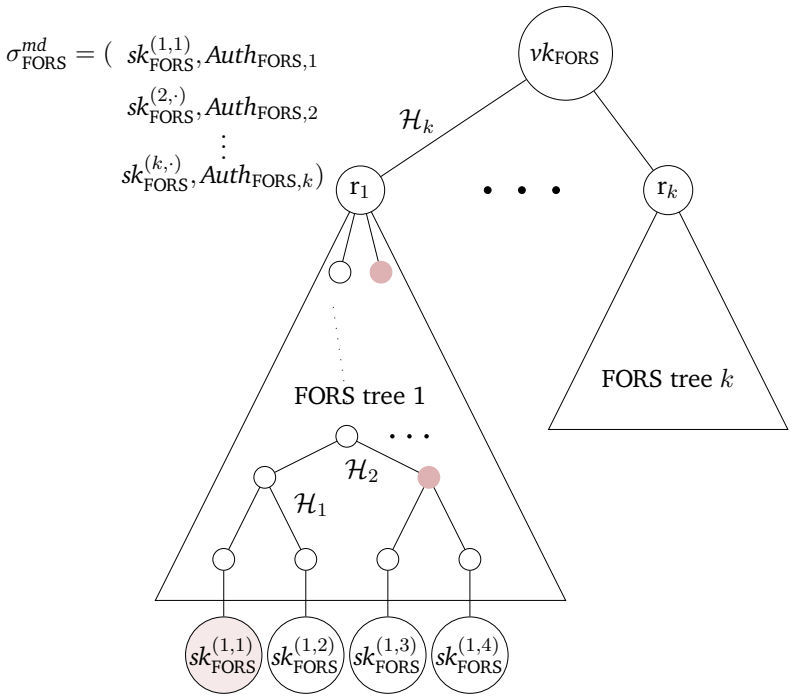


FIGURE 4.6: The forest of a FORS signature spanning k binary hash trees. Each signature is a collection of k a private key strings $sk^{\cdot,\cdot}_{\text{FORS}}$ along with the authentication path (shaded nodes) in the tree. All FORS trees are of equal size and simply scaled down for depiction.

EXTENDED MERKLE SIGNATURE SCHEME. An XMSS in SPHINCS$^+$ is a binary hash tree where each node is a λ bit string and the leaves are WOTS$^+$ public keys. Recall that h is the height of the SPHINCS$^+$ hypertree and d the number of layers in the hypertree. The height of each XMSS tree is $h' = h/d$, The tree is constructed using the hash functions $\mathcal{H}_1$ and $\mathcal{H}_2$.

The leaves of the tree are WOTS$^+$ public keys, where each tree has $2^{h'}$ leaves. The first node, i. e., the node on the lowest level in the hashtree, is the output of evaluating $\mathcal{H}_1$ on a WOTS$^+$ public key.

Key Generation The secret key is a seed that allows to generate all these WOTS$^+$ instances, which in turn allows to compute the hash tree. The public key $vk_{\text{XMSS},i}$ is the root of this hash tree.

Signature A signature contains a WOTS$^+$ signature along with an authentication path, i. e., the siblings on the path from the leaf through the binary hash tree required to compute the root. The string *idx* determines the WOTS$^+$ instance to be used for the signature.

Verification An XMSS scheme does not have a separate VERIFY function, as it only requires to compute the public key from the signature scheme,

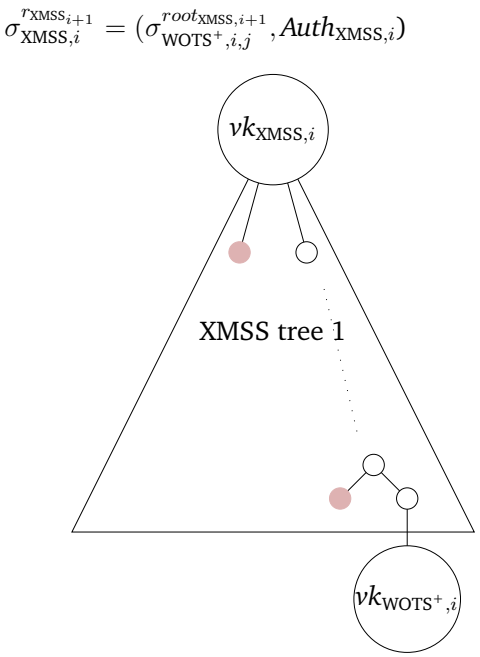


FIGURE 4.7: Binary hash tree of a XMSS scheme where the leaves are WOTS$^+$ public keys. The signature $\sigma^{r_{\text{XMSS},i+1}}_{\text{XMSS},i}$ contains a WOTS$^+$ signature of the root of another XMSS tree and an authentication path to compute the root node.

i. e., for the two topmost nodes p_1, p_2 in the binary tree before the root, the public key is computed as:

$$vk_{\text{SPHINCS}^+}.root \leftarrow \mathcal{H}(vk_{\text{SPHINCS}^+}.sd, ADRS, p_1, p_2) \ .$$

The WOTS$^+$ instances signs the root node of another XMSS instance. Figure 4.7 shows an explanatory XMSS tree.

HYPERTREE. The SPHINCS$^+$ [Hül+20, Sec. 4.2] scheme combines all signatures into one hypertree of total heigth h which has d layers of XMSS trees. Accordingly, top-most layer $d-1$ has one XMSS tree. The lowest layer, layer 0, has $2^{h-h'} = 2^{h-h/d}$ XMSS trees. Each XMSS tree is associated with $2^{h'}$ WOTS$^+$ instance. The instances used during a signing procedure is determined by the address *ADRS*.

Each XMSS tree signs the root node of a XMSS tree in the next layer. The lowest layers signs a FORS public key, which in turn signs a message digest.

[Hül+20] Hülsing et al., *SPHINCS+-Submission to the 3rd round of the NIST post-quantum project*

SPHINCS$^+$. The SPHINCS$^+$ signature scheme consists of a hyper tree of XMSS instances, WOTS$^+$ instances and FORS instances, using both keyed and tweakable hash functions. Figure 4.8 shows the complete SPHINCS$^+$ hyper tree with the individual signature schemes.
The exact algorithms of key generation, signing and verification are not important. We highlight the parts important to follow this manuscript:

Key Generation The public verification key is defined as $vk_{\text{SPHINCS}^+} = (vk_{\text{SPHINCS}^+}.sd, vk_{\text{SPHINCS}^+}.root)$ consisting of a seed and a root node. The secret key $sk_{\text{SPHINCS}^+} = (sk_{\text{SPHINCS}^+}.sd, sk_{\text{SPHINCS}^+}.\text{PRF}, vk)$ consist of a seed, the key of the PRF and the public key.

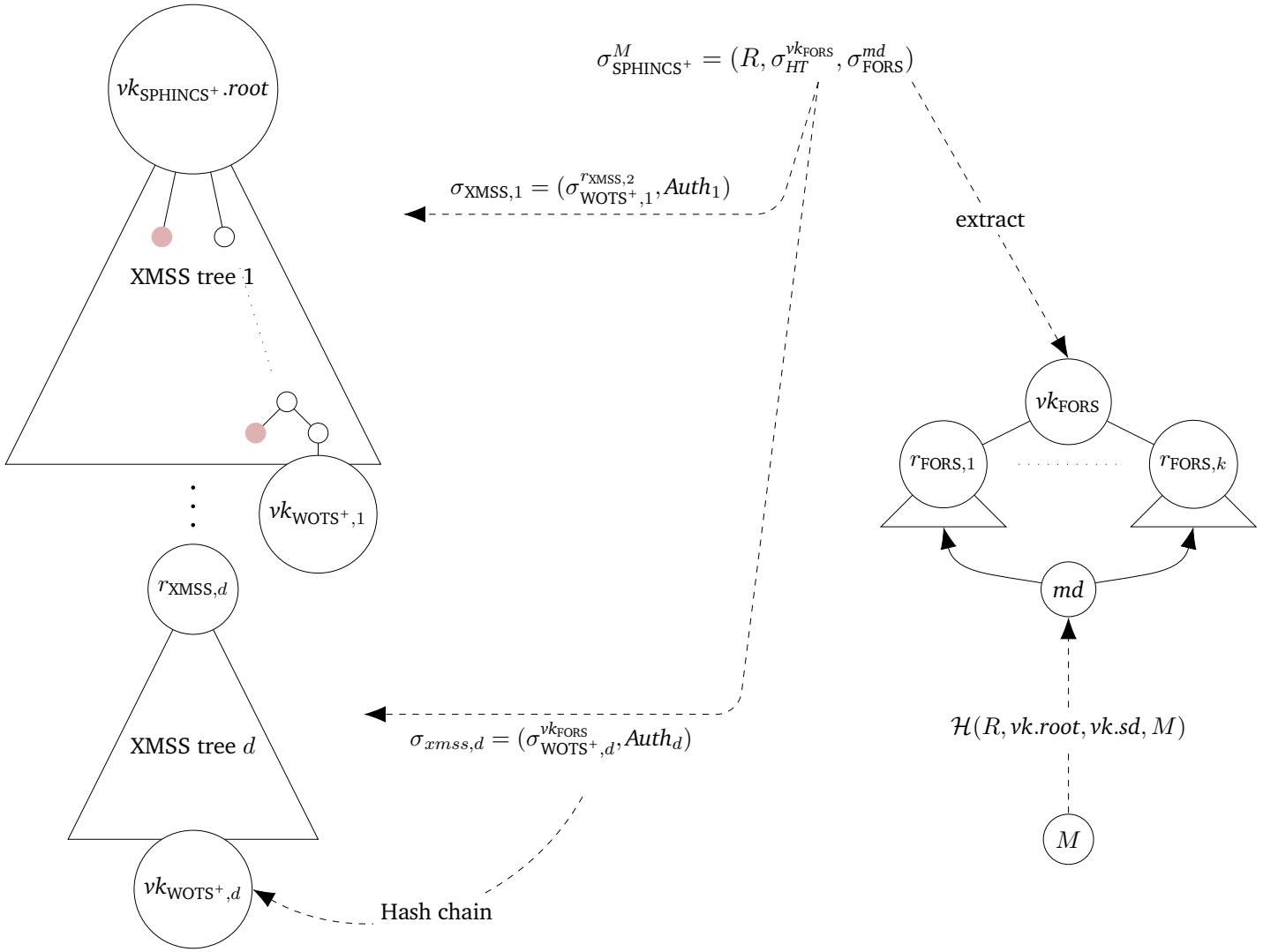


FIGURE 4.8: Overview of a SPHINCS$^+$ hypertree and a SPHINCS$^+$ signature $\sigma^{M}_{\text{SPHINCS}^+}$ on message M. The signature consists of a random string R used to generate an initial message digest, a collection $\sigma^{vk_{\text{FORS}}}_{HT}$ of XMSS signatures $\sigma_{\text{XMSS},i}$, and a FORS signature. Each XMSS signature $\sigma^{r_{\text{XMSS},i+1}}_{\text{XMSS},i}$ consists of a WOTS$^+$ signature σ_{WOTS^+} and of an authentication path $Auth_i$, the latter of which contains the siblings of the nodes in the path from the XMSS leaf to the respective root. The leaf of each XMSS tree, vk_{WOTS^+}, can be computed from the signature $\sigma^{r_{\text{XMSS}}}_{\text{WOTS}^+}$ by iterating the hash chain. The root of the FORS tree, vk_{FORS}, can be extracted from the FORS signature σ^{M}_{FORS}.

Signature A signature on a message M has the form

$$\sigma^{M}_{\text{SPHINCS}^+} = (R, \sigma^{vk_{\text{FORS}}}_{HT}, \sigma^{md}_{\text{FORS}}) ,$$

where $R \in \{0,1\}^\lambda$ is a string.

To generate a signature on a message M, first a message digest is computed:

$$md, id_{tree}, id_{leaf} \leftarrow \mathcal{H}_{msg}(R, vk_{\text{SPHINCS}^+}.sd, vk_{\text{SPHINCS}^+}.root, M) ,$$

where the values id_{tree}, id_{leaf} determine the FORS instance used to sign the message digest *md*. The hash function $\mathcal{H}_{msg}$ is the only compressing hash function in SPHINCS$^+$ that maps from an arbitrary length to a fixed length. Subsequently, (part of) the message digest *md* is signed using a FORS signature, $\sigma^{md}_{\text{FORS}}$, and the resulting public key vk_{FORS} is signed using the hypertree, resulting in $\sigma^{vk_{\text{FORS}}}_{HT}$.

Verification To verify a signature, first the message digest is computed from $\mathcal{H}_{msg}$, and then the FORS and hypertree signature are verified.

4.2.3 *Asymptotic Security*

The security of SPHINCS$^+$ is quantified based on the number of hash function invocations required to break the second preimage resistance. An estimate of the concrete security of SPHINCS$^+$ was given within the scope of the NIST submission. The authors considered general attacks [Hül+20, Sec. 9.3.1] on the distinct-function multi-target second-preimage resistance of the underlying hash functions. Let $q_\mathcal{H}$ be the number of hash function invocations. The success probability of a quantum adversary breaking the multi-target second preimage resistance of a hash function is bounded by [HRS16, Prop. 2], [HRS16, Table 1, Col. 1] $O\left(\frac{(q+1)^2}{2^\lambda}\right)$. For SPHINCS$^+$, this term is bounded by [Hül+20, Sec. 9.3.]

$$\Theta\left(\frac{(q_{arg}+1)^2}{2^{8\lambda}}\right) .$$

[Hül+20] Hülsing et al., *SPHINCS+-Submission to the 3rd round of the NIST post-quantum project*

[HRS16] Hülsing, Rijneveld, and Song, "Mitigating Multi-target Attacks in Hash-Based Signatures"

4.2.4 *Concrete Security*

The security of SPHINCS$^+$ is closely related to the security of the underlying hash functions, which is either SHA-256 $-$ 256, SHAKE-256-256 or Haraka512 [Hül+20, Sec. 9.1]:

The SHAKE-256 and SHA-256 variant of SPHINCS$^+$ both provide a sufficient amount of security for all 5 NIST security levels, SHAKE-256 claims 256 bit of security against classical second preimage attacks, if the output is "sufficiently long" [NIS15, Sec. 2.4]. An analysis of the security of SHAKE-256 has been given in [Amy+16], whose result is the main motivation for our work. They present a quantum circuit to implement a Grover search and attack the 256-bit preimage resistance of the SHA3-256 hash function and give concise and fault-tolerant estimates for the resources required to implement such a circuit: They claim that their circuit requires $2^{153.8}$ surface code cycles[3] using $2^{12.6}$ logical qubits, resulting in an overall requirement of

[NIS15] NIST, *SHA-3 standard: Permutation-based hash and extendable-output functions*

[Amy+16] Amy et al., "Estimating the Cost of Generic Quantum Pre-image Attacks on SHA-2 and SHA-3"

[3] A surface code cyle relates to the fault-tolerant cost of a quantum computer, Section 3.3.3

[4] A logic qubit cyle is the product of surface code cyles and qubits used by a quantum computer (cf. Section 6.1)

about $2^{166.4}$ logical-qubit-cyles[4] using 2^{128} black-box queries for a 256-bit preimage search. Their results may be adapted to estimate the work required to break the hash function for the SPHINCS$^+$ signature scheme. However, there is still considerable ambiguity on the specific construction to forge a signature.

Haraka features two variants, Haraka-128 and Haraka-256, both of which claim 256 bits of security against classical and 128 bits of security against quantum attackers [Köl+16, Sec. 2]. Note that SPHINCS$^+$-Haraka only achieves security level I, II (cf. Table 3.1), due to the capacity of the sponge construction in SPHINCS$^+$-Haraka using only 256 bits. Attacking the second-preimage-resistance as described in [Ber+11] only requires about $2^{129.5}$ classical hash function invocations, producing a collision on the internal state of the hash function in the process. The trade-off presented by [CNS17, Thm 2] results in about 2^{102} iterations for a collision search with $\lambda = 256$ on Haraka. The quantum security of Haraka has not, to the best of our knowledge, been analyzed explicitly.

[Köl+16] Kölbl et al., "Haraka v2 - Efficient Short-Input Hashing for Post-Quantum Applications"

[Ber+11] Bertoni et al., *Cryptographic sponge functions*

[CNS17] Chailloux, Naya-Plasencia, and Schrottenloher, "An Efficient Quantum Collision Search Algorithm and Implications on Symmetric Cryptography"

4.3 LATTICE-BASED CRYPTOGRAPHY

Lattice-based cryptography was introduced by Ajtai [Ajt96] in 1996, who presented the first worst-case to average-case reduction of lattice problems along with a first cryptosystem. The initial results were extended and improved, resulting in many different computational hardness assumption for both encryption and authentication presupposes. Noticeably, three out of four of NIST's finalists are based on lattice problems.

[Ajt96] Ajtai, "Generating hard instances of lattice problems (extended abstract)"

A lattice[5] $\mathcal{L} \subseteq \mathbb{R}^n$ is a discrete additive subgroup $(\mathbb{R}^n, +)$ generated by a set of linearly independent vectors $B = (b_1, \ldots, b_r) \in \mathbb{R}^{n \times r}, b_i \in \mathbb{R}^n$ called a basis. We denote B^* the result of performing Gram-Schmidt orthogonalization on B. The lattice is defined as $\mathcal{L} = \{\sum_{i=1}^{r} b_i c_i | c_i \in \mathbb{Z}^r\}$. The lattice is said to have dimension n and rank r. We call $c = (c_1, ..., c_r)^\top$ the coefficient vector of $\mathcal{L}$. All lattices in this work are full-rank integer lattices $\mathcal{L} \subseteq \mathbb{Z}^r$ with $n = r$. We denote $||v|| = \left(v_1^2 + ... + v_n^2\right)^{1/2}$ the euclidean norm of a vector. We let $\lambda_1(\mathcal{L}) = \min_{v \in \mathcal{L} \setminus \{0\}} ||v||$ denote the first minimum of the lattice.

[5] An in-depth introduction into lattices can be found here [Pei16].

4.3.1 *Computational Hardness Assumption*

One of the most central problems in the setting of lattice-based cryptography is the shortest vector problem from Definition 4.3.1.

Definition 4.3.1 (SVP)**.** *Given a basis B of a lattice $\mathcal{L}$, find the shortest non-zero vector $v \in \mathcal{L}$ such that $||v|| \leq \lambda_1(\mathcal{L})$.*

The approximate shortest vector problem, Shortest Vector Problem (SVP)$_\gamma$, is defined similarly, asking to find a vector that is γ-close to the first minimum, i. e., such that $||v|| \leq \gamma\lambda_1(\mathcal{L})$. Another problem is the (approximate) *decisional* (γ)-GapSVP problem, which asks to determine if the first minimum of the lattice is within a given bound r, i. e., $\lambda_1(\mathcal{L}) \leq \gamma r$. While the GapSVP problem was shown to be NP-hard [Ajt96] with a randomized reduction, the case for the approximate $\gamma-$GapSVP problem is a

bit more nuanced. Specifically, for $\gamma \in 2^{O(n)}$ the famous LLL algorithm (cf. Section 4.3.3) solves the problem in polynomial time, but it is NP-hard for any constant approximation $\gamma \in O(1)$. Figure 4.9 summarizes for which instantiations of γ the *approximate* GapSVP is believed to be difficult to solve.

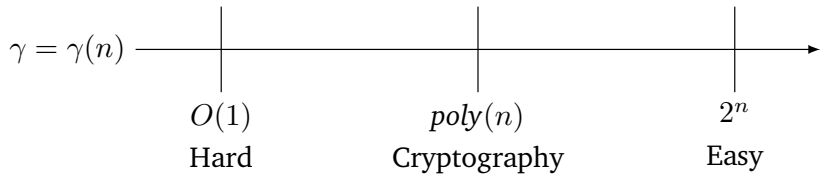


FIGURE 4.9: "Simplified summary of the complexity of the (γ-Gap)CVP on lattices of dimension n for constant, polynomial, and exponential approximation factor γ [...]." Figure rebuild and text *verbatim* from [Ben23, Fig. 1]. The values gives the size of γ.

Instead of building directly on a variant of the (gap) shortest vector problem, most cryptographic protocol build on the hardness of either search Learning with Errors (LWE) (c.f. Definition 4.3.2) or decision LWE (c.f. Definition 4.3.3), both of which can be reduced to a variant of the SVP.

Definition 4.3.2 (LWE Distribution)**.** *Let n_{lwe}, q, B be a positive integers and $\chi = [-B \dots B]$ define a distribution. Let $s \xleftarrow{\$} \mathbb{Z}_q^n$ and $e \xleftarrow{\$} \chi$ be sampled accordingly. Then we denote $A_{s,\chi}$ the distribution $(A, z_i := A^\top \cdot s + e) \in \mathbb{Z}_q^{n_{lwe}} \times \mathbb{Z}_q$.*

Definition 4.3.3 (LWE)**.** *Let n_{lwe}, q be positive integers and let (A, z_i) be sampled from $A_{s,\chi}$. The decision-LWE (dLWE) problem ask to distinguish samples from $A_{s,\chi}$ from the uniform distribution on $\mathbb{Z}_q^{n_{lwe}} \times \mathbb{Z}_q$. The search-LWE (sLWE) problem asks to find the vector s from the distribution A.*

We sketch a reduction from LWE to SVP in the form of the *primal attack* [APS15a; SB17; Alb+17]. The input is a set of LWE samples over the lattice used in the cryptosystem.

[APS15a] Albrecht, Player, and Scott, *On The Concrete Hardness Of Learning With Errors*

[SB17] Schmidt and Bindel, *Estimation of the Hardness of the Learning with Errors Problem with a Restricted Number of Samples*

[Alb+17] Albrecht et al., *Revisiting the Expected Cost of Solving uSVP and Applications to LWE*

[LM09] Lyubashevsky and Micciancio, "On Bounded Distance Decoding, Unique Shortest Vectors, and the Minimum Distance Problem"

[Kan87] Kannan, "Minkowski's Convex Body Theorem and Integer Programming"

1. The search version of LWE can be rephrased as a Bounded Distance Decoding (BDD) problem [LM09]: Given a lattice basis B and a vector z with the promise that z is close to lattice vector $v \in \mathcal{L}(B)$: $\text{dist}(z, B) < \alpha\lambda_1(B)$, where $\text{dist}(z, B)$ is the distance from z to the lattice generated by B, the goal is to find the vector v. In the setting of LWE, the lattice point z is close to a linear combination of columns of A, i. e., has a distance to $z = A^\top s$ bounded by the distribution χ.

2. The samples are embedded into a lattice that admits the target vector (usually the error e) as a short vector. Once the error vector e is known, $A^T s = z_i - e$ can to solved for s via Gaussian elimination. Given a number r of LWE samples $(A, z = As + e \mod q)$ they are embedded into a lattice that admits the vector $v = \binom{e}{t}$ as a short vector $\binom{-e}{t}$. A possible construction is *Kannan*'s Embedding [Kan87]:

$$B = \begin{pmatrix} A & z \\ 0 & t \end{pmatrix} \in \mathbb{Z}^{(r+1)\times(n_{lwe}+1)},$$

.

3. BDD in turn can be solved via reduction to the shortest vector problem. The embedding lattice is reduces using a lattice reduction algorithm, which in turn calls an SVP-solver as a subroutine on a sub-lattice, which are briefly reviewed in the following Section 4.3.3.

4.3.2 *Security*

The state-of-the-art attacks on these lattice problems usually involve the use of lattice reduction techniques, with block reduction algorithms being the most popular choice [SE94; CN11; Aon+16; MW16; Alb+19a; GN08a]. The leading cost of block lattice reduction (and therefore, often, of the attacks overall) comes from solving instances of the (approximate [LN20; Alb+21]) shortest vector problem (SVP) in high dimension.

The leading choice for (approximate) SVP solvers are lattice point enumeration [Kan83; FP85; GNR10; CN11; ANS18; Alb+20a] and sieving [AKS01; NV08; Laa15; Bec+16; Alb+19a] algorithms. Due to the central role these algorithms play in the cryptanalysis of lattice-based constructions [LP11; Alk+16; Alb+18] and because multiple post-quantum soon-to-be standards are lattice-based [Sch+22; Lyu+22; Pre+22], clearly understanding their cost is crucial. Enumeration and sieving are originally classical algorithms, with asymptotically different *classical* runtime (namely, $2^{O(n \log n)}$ for enumeration and $2^{O(n)}$ for sieving) and memory cost (namely, $O(poly(n))$ for enumeration and $2^{O(n)}$ for sieving).

4.3.3 *Lattice Reduction*

LLL. The algorithm of Lenstra-Lenstra-Lovász reduces an input basis $B = \{b_1, ..., b_r\}, b_i \in \mathbb{R}^n$ by alternating the Gram-Schmidt orthogonalization, reducing the vectors in sub-lattice and swapping vectors. It outputs a basis with $\frac{||b_i^*||}{||b_{i-1}||} \geq \delta - \mu_{i,i-1}^2$, where $\mu_{i,i-2} = \frac{b_i b_j^*}{b_j^* b_j^*}$ is the Gram-Schmidt coefficient. For $\delta \in (\frac{1}{4}, 1)$, it runs in time polynomial time in the lattice dimension and rank, specifically in time $O(r^5 \cdot n \log^3 \max_i(||b_i||_2))$. There haven been improvements over the run time, which are not relevant for this manuscript.

Informally this means that the LLL algorithm can find the *shortest* vector in a lattice (the first minimum) in polynomial time, if and only if, the gap to the next largest vector is sufficiently large [Boe+18]. For instance, this is used in the Slice-and-Dice attack (cf. Section 4.1.3). Otherwise, the runtime is exponential in the dimension, rank and approximation gap.

Remark 3. *The LLL algorithm was analyzed in the quantum regime by [TS19], showing how the algorithm can be implemented in a quantum circuit. The authors suggest that quantum implementations may suffer from a significant blow-up in required quantum memory [TS19, Eq. 8] that can be more than quadratic in the dimension n.*

BKZ. The Blockwise Korkine-Zolotarev Algorithm [SE94] reduces a given lattice basis block-wise by alternating calls to the LLL algorithm and an SVP oracle. The algorithm outputs a basis that is Hermite-Korkine-Zolotare reduced basis for each block of size β, and that admits, w.l.o.g., b_1 as a shortest vector. BKZ 2.0 [CN11] includes improvements of the underlying SVP-oracle such as pruning techniques and pre-processing of basis, as well

[SE94] Schnorr and Euchner, "Lattice Basis Reduction: Improved Practical Algorithms and Solving Subset Sum Problems"

[CN11] Chen and Nguyen, "BKZ 2.0: Better Lattice Security Estimates"

[Aon+16] Aono et al., "Improved Progressive BKZ Algorithms and Their Precise Cost Estimation by Sharp Simulator"

[MW16] Micciancio and Walter, "Practical, Predictable Lattice Basis Reduction"

[Alb+19a] Albrecht et al., "The General Sieve Kernel and New Records in Lattice Reduction"

[GN08a] Gama and Nguyen, "Finding short lattice vectors within Mordell's inequality"

[LN20] Li and Nguyen, *A Complete Analysis of the BKZ Lattice Reduction Algorithm*

[Alb+21] Albrecht et al., "Lattice Reduction with Approximate Enumeration Oracles - Practical Algorithms and Concrete Performance"

[Sch+22] Schwabe et al., *CRYSTALS-KYBER*

[Lyu+22] Lyubashevsky et al., *CRYSTALS-DILITHIUM*

[Pre+22] Prest et al., *FALCON*

[Boe+18] Boer et al., "Attacks on the AJPS Mersenne-Based Cryptosystem"

[TS19] Tiepelt and Szepieniec, "Quantum LLL with an Application to Mersenne Number Cryptosystems"

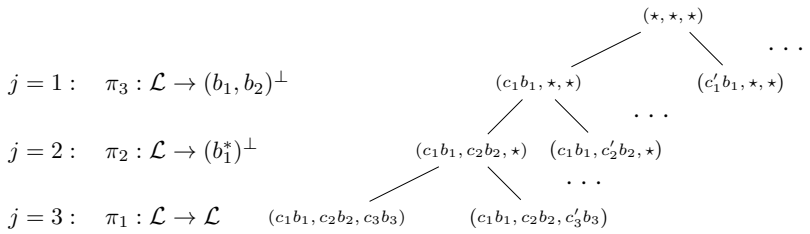


FIGURE 4.10: Simplified example of an lattice enumeration backtracking tree for a lattice with rank $r = 3$. Let the lattice vectors be of the form $v = \sum_{i=1}^{r} c_i b_i$. Then every node in the tree on level j is in the projective sublattice $\pi_j(\mathcal{L})$. Each such node corresponds to a assignment of values to $c_i \in \mathbb{Z}$ for $1 \leq i \leq j$.

as a heuristic for an early abort for BKZ. The runtime of the algorithm is dominated by the cost of the SVP oracle, an algorithm that finds a *sufficiently* short vector in a lattice. Such an oracle is usually instantiated with either lattice sieving or lattice enumeration, the latter of which we cover in the next Section 4.3.4.

4.3.4 *Lattice Enumeration*

Lattice enumeration is the procedure of systematically iterating all points in a lattice; if one considers *short* lattice enumeration, then the space is bounded, i. e., one is searching for all lattice vectors with length at most R. Lattice enumeration works by performing a depth-first-search over projections (cf. Definition 4.3.4) of lattice vectors. The depth-first-search implicitly constructs an enumeration tree as in Figure 4.10 analog to a backtracking tree (cf. Definition 3.1.1), where the variables correspond to coefficients in the projected lattice. As originally proposed, enumeration algorithms iterated over a tree spanning the complete intersection of the lattice with a ball of radius R around the origin [Poh81; Kan83; FP85; SE94]. Modern variants restrict the search space by introducing a branch-and-bound methodology, pruning the tree based on a heuristic bound on the norm $||\pi_i(v)||$ of the target's orthogonal projections [GNR10; MW15; Alb+20a]. This slightly reduces the success probability p of the algorithm, since the short vector may be erroneously pruned from the tree, introducing a trade-off between faster traversal speed on the smaller tree versus having to re-run the procedure $O(p^{-1})$ times on re-randomised versions of the lattice basis. In this manuscript, we focus on the extreme cylinder pruning variant originating from [SE94, Alg. ENUM] used in [GNR10], with pruning bounds R_k for each level k of the search tree.

[Poh81] Pohst, "On the Computation of Lattice Vectors of Minimal Length, Successive Minima and Reduced Bases with Applications"

[Kan83] Kannan, "Improved Algorithms for Integer Programming and Related Lattice Problems"

[FP85] Fincke and Pohst, "Improved methods for calculating vectors of short length in a lattice, including a complexity analysis"

[SE94] Schnorr and Euchner, "Lattice Basis Reduction: Improved Practical Algorithms and Solving Subset Sum Problems"

[GNR10] Gama, Nguyen, and Regev, "Lattice Enumeration Using Extreme Pruning"

[MW15] Micciancio and Walter, "Fast Lattice Point Enumeration with Minimal Overhead"

[Alb+20a] Albrecht et al., "Faster Enumeration-Based Lattice Reduction: Root Hermite Factor $k^{1/(2k)}$ Time $k^{k/8+o(k)}$"

PROCEDURE. Given an input basis B of $\mathcal{L}$ and an upper bound $\lambda_1(\mathcal{L}) \leq R$ (e. g., $R = ||b_1^*||$), lattice enumeration finds a vector $v \in \mathcal{L}$ such that $||v|| \leq R$. The central projection map is given in Definition 4.3.4. The projection $\pi_i(\mathcal{L})$ is a $r - i + 1$ dimensional lattice. Given a lattice $\mathcal{L}$ of rank r, the set $\pi_i(\mathcal{L}) = \{\pi_i(v) \mid v \in \mathcal{L}\}$ is itself a lattice in $\mathbb{R}^{r-i+1}$. We sometimes refer to such lattices as *projective sublattices*.

Definition 4.3.4 (Projection π_i)**.** *Let B be a basis for $\mathcal{L}$. The projection*

$$\pi_i : \mathcal{L} \to \mathit{span}(b_1, ..., b_{i-1})^\perp ,$$

maps lattice points onto the space orthogonal to the span of vectors $b_1, ..., b_{i-1}$.

The procedure performs a depth-first-search on a tree consisting of an "empty node" root on level $k = 0$, and of a set of projected lattice points of norm at most R in $\pi_{r-k+1}(\mathcal{L})$ on level $k > 0$. The leaves of the tree on level $k = r$ are lattice points of norm at most R.

The key observation behind lattice enumeration algorithms is that orthogonal projections cannot increase the norm of a vector. This means, for a lattice $\mathcal{L}$ with basis $B = (b_1, \ldots, b_r)$, shortest vector v, and sufficiently large Rthat $R \geq ||v|| = ||\pi_1(v)|| \geq ||\pi_2(v)|| \geq \cdots \geq ||\pi_r(v)|| \geq 0$, with $\pi_i(v)$ living in an $(r-i+1)$-dimensional subspace of $\mathbb{R}^r$. Thus, to enumerate vectors in $\mathcal{L}$ of norm at most R, it is sufficient to enumerate vectors in the lower-rank projections $\pi_i(\mathcal{L})$ for $i \leq r$, discarding guesses for $\pi_i(v)$ if they are too long.

Starting from $i = r$, suppose $\pi_r(v) = g_r$ is guessed correctly for some vector $g_r \in \pi_n(\mathcal{L})$.[6] The enumeration algorithm then attempts to extend this into a guess $g_{r-1} \in \pi_{n-1}(\mathcal{L})$ for $\pi_{r-1}(v)$ such that $\pi_r(g_{r-1}) = g_r$. If $||g_{r-1}|| \leq R$,[7] one proceeds similarly trying to extend g_{r-1} into a guess for g_{r-2} for $\pi_{r-2}(v)$ and so on; else one attempts to find a different guess $g'_{r-1} \neq g_{r-1}$ for $\pi_{r-1}(v)$ that is short enough, and if no such vector exists one aborts the search in $\pi_{r-1}(\mathcal{L})$ and attempts to extend a different guess $g'_r \neq g_r$ for $\pi_r(v)$. Every guess g_i for $\pi_i(v)$ of norm at most R becomes a node in the enumeration tree. A node g_i is the *child* of some guess g_{i+1} for $\pi_{i+1}(v)$ such that $\pi_{i+1}(g_i) = g_{i+1}$. Moreover, every node g_i is the *parent* of guesses $\{g_{i-1} \in \pi_{i-1}(\mathcal{L}) \mid \pi_i(g_{i-1}) = g_i\}$ for $\pi_{i-1}(v)$.

Careful computation using the Gram-Schmidt vectors $B^* = (b_1^*, \ldots, b_r^*)$ and coefficients $\mu_{i,j} = b_i \cdot b_j^*/||b_j^*||^2$ for $i > j$, shows that given a lattice vector $v = \sum_{i=1}^r c_i b_i$ where $c_i \in \mathbb{Z}$ for all i, its projections are of the form $\pi_j(v) = \sum_{i=j}^r \alpha_i b_i^*$ where $\alpha_j = c_j + \sum_{i>j} \mu_{i,j}\, c_i$. By orthogonality of the $(b_i^*)_i$, we have $||\pi_j(v)||^2 = |\alpha_j|^2||b_j^*||^2 + ||\pi_{j+1}(v)||^2$. Hence, for any guess g_{j+1} for $\pi_{j+1}(v)$, a guess g_j for $\pi_j(v)$ with $\pi_{j+1}(g_j) = g_{j+1}$ must satisfy $|\alpha_j|^2 \leq (R^2 - ||\pi_{j+1}(v)||^2)/||b_j^*||^2$, *i.e.*,

$$\begin{aligned} \left|c_j + \sum_{i>j} \mu_{i,j}\, c_i\right| &\leq \frac{\sqrt{R^2 - ||\pi_{j+1}(v)||^2}}{||b_j^*||} \\ &= \frac{\sqrt{R^2 - \sum_{r=j+1}^{r} \left(c_r + \sum_{i=r+1}^{r} \mu_{i,r} c_i\right)^2 ||b_r^*||^2}}{||b_j^*||}. \end{aligned} \tag{4.4}$$

Remark 4. *In the case of pruned enumeration, the main difference in the process is that we are given a* pruning set $\{R_i \mid i \in [n], 0 < R_1 \leq \cdots \leq R_n = R\}$ *rather than a single bound R, with R_{n-j+1} replacing R in Equation (4.4).*

EXPECTED COST OF ENUMERATION. The cost of enumeration is typically estimated to be equal to the number of nodes visited by the algorithm—the "enumeration tree" $\mathcal{T}$ of size $\#\mathcal{T}$. Let Z_k be the set of nodes on the k^{th} level of the tree (that is, at distance k from the empty root node), H_k be the expected cardinality of Z_k over the distribution of random bases being enumerated,[8] and N be the total number of nodes in the tree. The cardinality of Z_k depends on the pruning strategy and on the geometry of the projective

[6] Since lattices are symmetrical around the origin, in practice implementations consider only half of the possible guesses for $\pi_i(v)$.

[7] To unburden notation, we temporarily consider the non-pruned case with $R_k = R$, $\forall k$.

[8] In the case of BKZ reduction with extreme cylinder pruning, these are re-randomized instances of a local BKZ block.

sublattices implied by the lattice bases. The expected number of total nodes is $\mathbb{E}[\#\mathcal{T}] = \frac{1}{2}\sum_{k=1}^{n} \mathbb{E}[|Z_k|] = \frac{1}{2}\sum_{k=1}^{n} H_k$, where the expectation is taken over the distribution from which the lattice is being sampled and the extreme cylinder pruning re-randomization (if any is used), and the $1/2$ factor is due to exploiting lattices' additive symmetry to avoid unnecessarily visiting half of the tree. Cost estimation for the algorithm then reduces to estimating H_k. This is a standard computation that we perform in detail in [Bin+23, App. A] closely following [GNR10; Aon+18]. However, the exact procedure is not relevant to understand the content of this manuscript.

[Bin+23] Bindel et al., *Quantum Lattice Enumeration in Limited Depth*

[GNR10] Gama, Nguyen, and Regev, "Lattice Enumeration Using Extreme Pruning"

[Aon+18] Aono et al., "Lower Bounds on Lattice Enumeration with Extreme Pruning"

[ANS18] Aono, Nguyen, and Shen, "Quantum Lattice Enumeration and Tweaking Discrete Pruning"

Quantum Speedups. Aono, Nguyen and Shen [ANS18] analyze the asymptotic cost of using quantum backtracking algorithms to perform lattice enumeration in a black-box setting. In [ANS18, Thm. 7(1)], they identify the asymptotic runtime for finding a short non-zero vector using cylinder pruning and poly(n) qubits to be $O\left(\sqrt{\#\mathcal{T}}n^3 \text{ poly}(\log(1/\delta), \log n)\right)$. In [ANS18, Sec 4.2], they argue that this holds also when performing extreme pruning with M randomized lattice bases $(B_i)_{i\leq M}$, by collecting each enumeration tree into a larger, single tree, resulting in an asymptotic runtime for finding a non-zero vector via quantum extreme pruning of $O\left(\sqrt{\#\mathcal{T}^M}n^3 m \text{ poly}(\log n, \log(1/\delta), \log(m), \log M)\right)$, where m is the bitsize of the entries of B_i, for $i \in [M]$ and $\#\mathcal{T}^M$ is the sum of the number of nodes of all M trees.

While applicability of these speedups appears clear in the unbounded-depth logical-qubit model, where quantum computation achieves low error rates for free and does not decohere, our current understanding of quantum computer engineering suggests that this model may be overly optimistic for hypothetical real-world quantum adversaries [Pre18]. For example, Albrecht, Gheorghiu, Postletwaite and Schanck [Alb+20b] investigate the impact of error correction on quantum lattice sieving, determining that achieving even small speedups over classical sieving in the cryptanalytic regime requires making several optimistic algorithmic and physical assumptions.

[Pre18] Preskill, "Quantum Computing in the NISQ era and beyond"

[Alb+20b] Albrecht et al., "Estimating Quantum Speedups for Lattice Sieves"

5

Exploiting Decryption Failures

In this chapter we explore how *decryption failures* can be exploited to break the IND-CCA security of Mersenne-number cryptosystems, two of which, Ramstake [Sze17] and Mersenne-756839 [Agg+17b], were submitted to the first round of the NIST post-quantum competition.

Parts of this chapter are taken verbatim from our publications [TD20b; TD20a].

[Sze17] Szepieniec, *Ramstake*

[Agg+17b] Aggarwal et al., *Mersenne-756839*

Decryption Failures. A decryption failure occurs when an attempt to decrypt encrypted data is unsuccessful. As these decryption failures also depend on the secret key, they contain information on the secret key. It is important to note that decryption failures are not always induced by malicious activity; they can also occur due to technical issues, such as the incorporation of error correcting codes in the encryption and decryption process.

Various proposals in the NIST post-quantum competition [Nat20] are prone to a low, but non-zero probability that ciphertexts fail to decrypt correctly, for which the two communicating parties fail to agree on a common message after the execution of the protocol. These failures occur in the Mersenne number schemes such as Ramstake [Sze17] (with failure probability 2^{-64}) or Mersenne-756839 (with failure probability 2^{-239}) [Agg+17b]. Likewise many other NIST proposals admit such failures, as in the family of lattice based (e.g. FrodoKEM [Nae+17] with 2^{-252}, Kyber [Bos+17] with 2^{-160}, Saber [DVV18] with 2^{-136}) or code based schemes (e.g. HQC [Mel+18] with 2^{-128}, LEDAcrypt [Bal+18] with 2^{-64}, or Rollo [Mel+19] with 2^{-42}).

[Nat20] National Institute for Standards and Technology, *Post-Quantum Cryptography Round 1*

[Nae+17] Naehrig et al., *FrodoKEM*

[Bos+17] Bos et al., *CRYSTALS – Kyber: a CCA-secure module-lattice-based KEM*

[DVV18] D'Anvers, Vercauteren, and Verbauwhede, *On the impact of decryption failures on the security of LWE/LWR based schemes*

[Mel+18] Melchor et al., *Hamming Quasi-Cyclic (HQC)*

[Bal+18] Baldi et al., *LEDAcrypt: Low-dEnsity parity-chck coDe-bAsed cryptographic systems*

[Mel+19] Melchor et al., *ROLLO-Rank-Ouroboros, LAKE & LOCKER*

For lattice based schemes, Jaulmes and Joux [JJ00] introduced a chosen ciphertext attack leveraging decryption failures, which was later refined and extended by Gamma, Nguyen and Howgrave-Graham [How+03]. These attacks are countered by schemes that obtain IND-CCA security using an appropriate transformation. At the same time, D'Anvers et al. [DAn+19a] provided a technique to increase the failure probability and subsequently recover the secret key of IND-CCA secure LWE based schemes, a technique which was extended in subsequent works [DVV19; GJY19; DRV20]. Guo, Johansson and Stankovski [GJS16] provided a similar attack on IND-CCA secure code based schemes.

[JJ00] Jaulmes and Joux, "A Chosen-Ciphertext Attack against NTRU"

[How+03] Howgrave-Graham et al., "The Impact of Decryption Failures on the Security of NTRU Encryption"

[DAn+19a] D'Anvers et al., "Decryption Failure Attacks on IND-CCA Secure Lattice-Based Schemes"

[GJS16] Guo, Johansson, and Stankovski, *A Key Recovery Attack on MDPC with CCA Security Using Decoding Errors*

Objective & Contribution. Naturally, one may ask how or if Mersenne number cryptosystems are affected by these techniques. We aim to understand, what impact decryption failures have on the security of Mersenne-

based key encapsulation mechanisms. That means to understand how much information can be extracted from one or multiple decryption failures, what the nature of that information is, and how this impacts the secrecy of the secret key, or otherwise affects the security notions.

We present a novel method on exploiting decryption failures of Mersenne number cryptosystems [Agg+17b; Sze17], both of which were candidates in the first round of the NIST post-quantum competition [Nat20]. Particularly, we present an attack exploiting this information to break the IND-CCA security of Mersenne number cryptosystems.

[Agg+17b] Aggarwal et al., *Mersenne-756839*

[Sze17] Szepieniec, *Ramstake*

[Nat20] National Institute for Standards and Technology, *Post-Quantum Cryptography Round 1*

Our attack provides a *good* estimate of the secret keys when given enough failing ciphertexts, which allows to circumvent the bottleneck of the cost of the Slice-n-Dice attack (cf. Section 4.1.3) introduced by Beunardeau et al. [Beu+19] to extract these secrets. This effectively allows to break the IND-CCA security of the cryptosystems in question as well as extract the secret key, Specifically, we show that Proposition 1 holds true.

[Beu+19] Beunardeau et al., "On the Hardness of the Mersenne Low Hamming Ratio Assumption"

Proposition 1 (Partition from Decryption Failures; Informal)**.** *Decryption failures in the Mersenne number cryptosystems [Agg+17b; Sze17] allow to estimate the positions of ones in the secret vectors sufficiently well to provide a partitioning for the Slice-and-Dice attack.*

We provide empirical results for attacking the Ramstake cryptosystem, which can be reproduced and verified using our implementation. The empirical results obtained good estimates of the secret using 2^{12} decryption failures. These can be extracted from the original Ramstake scheme in about 2^{74} decapsulations queries. The original attack of the Slice-and-Dice attack requires 2^{256} classical guesses, or 2^{128} quantum invocations of a guessing function to extract the secret keys. We will show that our cryptanalytic approach is able to identify the positions of the secrets sufficiently good to reduce the cost of the Slice-and-Dice attack to require 2^{46} iterations of Grover's algorithm, each of which corresponds to one quantum invocation of the guessing function, to extract the secret key.

OUTLINE. In Section 5.1 we review the Mersenne prime schemes Ramstake [Sze17] and Mersenne-756839 [Agg+17b] along with their potency to generate decryption failures. In Section 5.2 we define heuristics that allow us to quantify the probability distribution of the secrets. Based on these heuristics we present our method on estimating the secret vectors. We introduce an estimator for the bits of the secret key using decryption failures, such that on input of a set of decryption failures we get an estimate on the probability distribution of the positions of the ones in the secret keys. In Section 5.3, we show that our estimates allow to derive additional knowledge about the secret key and how that information can be used to speed up the attack by Beunardeau et al. significantly, providing a *new* upper bound on the complexity to break IND-CCA security of Mersenne schemes given a set of decryption failures. Finally, we report on our implementation and the results of the attack on the Ramstake cryptosystem.

5.1 DECRYPTION FAILURES IN MERSENNE-BASED SUBMISSIONS TO NIST

Both Mersenne-756839 and Ramstake have a small probability of decryption failures, in which the keys are not transmitted correctly. In the generic Mersenne scheme, Figure 4.2 of Section 4.1 errors are introduced into the ciphertext by the xor operations with consecutively $S[0 : |r_{ecc}|]$ and $S'[0 : |r_{ecc}|]$. A decryption failure occurs, if these errors cannot be corrected by the error correcting code.

To ease the work on decryption failures and bitstrings in the ring modulo a Mersenne number p, we use following notation throughout this section: Counting the Hamming weight of the substring $x[i : j]$ will be abbreviated as $\text{HW}_{[i:j]}(x)$ and likewise $\text{HW}^8_{[i:j]}(x)$ for bytewise Hamming weights. Given two integers $x, y \in \mathbb{Z}_p$ the xor operation $\oplus$ will be defined so that $z = x \oplus y$ if $z[i] = (x[i] + y[i]) \mod 2$ for all $i \in [0, n)$.

5.1.1 *Mersenne-756839*

The Mersenne-756839 KEM [Agg+17b] implements a repetition code to instantiate the algorithms ENCODE and DECODE in the encryption and decryption algorithms (cf. Figure 4.2), where each bit of the message M is repeated $\chi = 2048$ times. During the encryption an additional error term $\hat{d}$ is added to the (shared) noisy secret resulting in an overall error of

$$(acG + ad) \oplus (acG + bc + \hat{d}) .$$

A decryption failure occurs if more than $\chi/2$ bits of any single encoded bit are erroneous.

[Agg+17b] Aggarwal et al., *Mersenne-756839*

5.1.2 *Ramstake*

The Ramstake KEM [Sze17] employs an error correcting code based on repetitive Reed-Solomon encodings as described in Figure 5.1. The Reed-Solomon[1] code maps a 256-bit message onto a 2040-bit (255 byte) codeword, which can correct up to t byte errors. We will denote the bit length of the codeword with l_c. This codeword is repeated ν times. Furthermore, a hash h of the message is included in the ciphertext. During decoding, the Reed-Solomon codes are decrypted iteratively and then checked for correctness by comparing the hashed value $\mathcal{H}(r)$ with h. More information on the design of Ramstake can be found in [Sze17], the parameters of the Ramstake instantiations can be found in Table 5.1. In this paper we will focus on the high security variant Ramstake-756839.

A failure occurs when none of the ν codewords could be decoded. The k^{th} codeword cannot be decoded if the bytewise Hamming weight $\text{HW}^8_{[kl_c:(k+1)l_c]}(e) > t$ is larger than the threshold t, where

$$e = S \oplus S' = ((acG + ad) \oplus (acG + bc)) ,$$

is the bitwise XOR of the shared noisy one-time-pads.

[Sze17] Szepieniec, *Ramstake*

[1] The exact working of Reed-Solomon codes is not relevant for us.

TABLE 5.1: Parameter sets for the two security levels of Ramstake. The highlighted row is the primary target of this chapter.

	p	ω	ν	t	$\mathbb{P}[F]$	security
Ramstake-216091	216091	64	4	111	$\leq 2^{-64}$	128
Ramstake-756839	756839	128	6	111	$\leq 2^{-64}$	256

RAMSTAKE.ENCODE(m)

1 : $e = \texttt{enc}_{RS}(m)$
2 : $m_{ecc} = 0$
3 : **for** $i = 0$ **to** $\nu - 1$
4 : $\quad m_{ecc} \mathrel{+}= e \cdot 2^i \frac{l}{\nu}$
5 : $\quad h = \mathcal{H}(m)$
6 : **return** m_{ecc}, h

RAMSTAKE.DECODE(m_{ecc}, h)

1 : **for** $i = 0$ **to** $\nu - 1$
2 : $\quad m = \texttt{dec}_{RS}\left(m_{ecc}[i\frac{l}{\nu} : (i+1)\frac{l}{\nu} - 1]\right)$
3 : $\quad$ **if** $h == \mathcal{H}(m)$
4 : $\quad\quad$ Return m
5 : **return** $\perp$

FIGURE 5.1: Error correcting encoding and decoding for Ramstake where $\texttt{enc}_{RS}$ and $\texttt{dec}_{RS}$ correspond to encoding and decoding of Reed-Solomon codes respectively.

5.2 FAILURE ATTACK

In this following section, we will focus our attention to Ramstake. The Ramstake KEM [Sze17] has a failure probability of 2^{-64}. That means, that after an expected number of 2^{64} *valid* queries a decryption failure will occur. We assume that the adversary has obtained N decryption failures, allowing them to estimate the probability of the bits in the secrets being *zero* or *one*. The probability is defined formally in Corollary 1. Later, in Section 5.3, we analyze the cost the adversary has to obtain the N decryption failures.

[Sze17] Szepieniec, *Ramstake*

In order to mount the failure attack, we first identify properties of error bits in Section 5.2.1 and then define a maximum likelihood estimation of the secret. This step takes as input a list of decryption failures, and outputs an estimation of the probability that each of the bits in the interval $[1, p]$ is zero or one. Each of the N decryption failures has corresponding integers $(c^{(j)}, d^{(j)}), 1 \leq j \leq N$ which were chosen by the adversary, and are corresponding to an unknown (to the adversary) but fixed secret key (a, b).

A failure indicates that all ν codewords in a ciphertext are decoded incorrectly. We will denote the number of errors in the k^{th} codeword of the j^{th} ciphertext with $F_{j,k}$, where

$$F_{j,k} = \text{HW}^8_{[kl_c:(k+1)l_c]}\left((ac^{(j)}G + ad^{(j)}) \oplus (ac^{(j)}G + bc^{(j)})\right),$$

so that a codeword is incorrectly decoded if $F_{j,k} > t$.

In the following derivation, we will first show that $\text{HW}^8_{[kl_c:(k+1)l_c]}(2^i b_i c^{(j)})$, is a reasonable indicator for the value of $F_{j,k}$ when one only has knowledge of $c^{(j)}, d^{(j)}$ and b_i. Then, we will use this to construct a maximum likelihood estimator to estimate the probability of the bits of a and b.

5.2.1 *Maximum likelihood estimation*

PROPERTIES OF THE ERROR BITS. We will assume that $b_i = x$ and that we know $c^{(j)}, d^{(j)}$. We will split the value of $F_{j,k}$ into a term that depends on b_i and a term with no dependency on b_i.

TABLE 5.2: Probabilities of values of $\text{HW}^8_{[kl_c:(k+1)l_c]}(2^i b_i c)$, the number of non-zero bytes, computed using Equation (5.1) and the parameters for Ramstake-756839 from Table 5.1.

HW^8	0	1	2	3	4	5
$\mathbb{P}$	70.82%	24.45%	8.41%	2.88%	0.98%	0.34 %

Heuristic 5.2.1. *The number of errors $F_{j,k}$ for a uniform random $G \leftarrow \mathcal{U}(\mathbb{Z}_p)$ and low Hamming weight $(a, b, c, d) \leftarrow \mathcal{HW}_\omega(\mathbb{Z}_p)$ is approximately the same as the sum of errors for $(0, 2^i b_i, c, d)$ and $(a, b - 2^i b_i, c, d)$, or:*

$$\begin{aligned} F_{j,k} &= \text{HW}^8_{[kl_c:(k+1)l_c]}\left((acG + ad) \oplus (acG + bc)\right) \\ &\approx \begin{pmatrix} \text{HW}^8_{[kl_c:(k+1)l_c]}(2^i b_i c) \\ + \text{HW}^8_{[kl_c:(k+1)l_c]}\left((acG + ad) \oplus (acG + (b - 2^i b_i)c)\right) \end{pmatrix} \end{aligned}$$

We justify the heuristic as follows: one can easily see that for $b_i = 0$, this heuristic is exact. For $b_i = 1$, the heuristic is exact if none of the nonzero bytes in $(2^i b_i c)$ coincides with the nonzero bytes of $\left((acG + ad) \oplus (acG + (b - 2^i b_i)c)\right)$ in the range $[kl_c : (k + 1)l_c]$. In the following reasoning, we will estimate the distribution of HW^8 for both terms, from which we will argue that overlaps are rare:

(1) The number of nonzero bits of $(2^i b_i c)$ is 128 out of 756,839 bits, which results in the probability of a non-zero byte as

$$\begin{aligned} \mathbb{P}[\text{bit zero}] &= 1 - \mathbb{P}[\text{bit non-zero}] = 1 - \frac{\omega}{p} \\ \mathbb{P}[\text{byte non-zero}] &= 1 - \mathbb{P}[\text{bit zero}]^8 = 1 - \left(1 - \frac{\omega}{p}\right)^8 . \end{aligned}$$

Accordingly, the probability that i out of 255 bytes are non-zero is

$$\begin{aligned} &\mathbb{P}[\text{HW}^8_{[kl_c:(k+1)l_c]}(2^i b_i c) = i] \\ &= \prod_{i=0}^{255} (255 - i) \cdot \mathbb{P}[\text{byte non-zero}] \cdot (1 - \mathbb{P}[\text{byte non-zero}])^{255-i} , \end{aligned} \tag{5.1}$$

which results in an average byte Hamming weight of under 0.373 bytes per codeword. The concrete probability for the distribution of Hamming weights is presented in Table 5.2.

(2) We estimated[2] the average byte Hamming weight of $\left((acG + ad) \oplus (acG + (b - 2^i b_i)c)\right)$ empirically by generating 1024 samples and obtained an average Hamming weight of 80.68, meaning that on average 80.68 out of 255 bytes, or approximately 31.64%, are erroneous.

[2]The estimation can be verified via the script "prob_non_zero_bytes.py".

The probability of one collision, and thus an error of one in our heuristic, can then be approximated as the probability of having a certain number of

bits in the first term, times the probability of a collision due to the second term, which can be made explicit as:

$$0.31 \cdot \sum_{i=1}^{255} \mathbb{P}\left[\text{HW}^8_{[kl_c:(k+1)l_c]}(2^i b_i c) = i\right] \approx 11.779$$

This gives an error in about 11.8% of the cases. However, the heuristic will only be off with a small number.

Heuristic 5.2.2. *For estimating $F_{j,k}$ calculated using a uniform random $G \leftarrow \mathcal{U}(\mathbb{Z}_p)$ and low Hamming weight $(a, b, c, d) \leftarrow \mathcal{HW}_\omega(\mathbb{Z}_p)$, knowledge of the tuple $(0, 2^i b_i, c, d)$ is as good as knowledge of the Hamming weight $\text{HW}^8_{[kl_c:(k+1)l_c]}(2^i b_i c)$, or:*

$$\begin{aligned}&\mathbb{P}\left[F_{j,k} \mid b_i = x, \{c^{(j)}, d^{(j)}\}_{j=1..N}\right]\\ \approx&\mathbb{P}\left[F_{j,k} \mid \text{HW}^8_{[kl_c:(k+1)l_c]}(2^i x c)\right]\end{aligned}$$

Following Heuristic 5.2.1, $F_{j,k}$ can be split in two parts:

$$\begin{aligned}&\text{HW}^8_{[kl_c:(k+1)l_c]}(2^i b_i c)\\ \text{and } &\text{HW}^8_{[kl_c:(k+1)l_c]}((acG + ad) \oplus (acG + (b - 2^i b_i)c))\,.\end{aligned}$$

Information about the tuple $(0, 2^i b_i, c, d)$ can be used to fully determine the first part, while the latter part has an unknown term a or $b - 2^i b_i$ in each of its terms. We argue that for this reason the tuple $(0, 2^i b_i, c, d)$ does contain negligible information about the second term. From this assumption follows that knowledge of the tuple $(0, 2^i b_i, c, d)$ gives the same information as knowledge about the Hamming weight $\text{HW}^8_{[kl_c:(k+1)l_c]}(2^i b_i c)$.
While these heuristics are clearly not exact, we will see that they are sufficiently close for our purposes.

ESTIMATOR. We will derive an estimator for the probability that the i^{th} bit of b equals x, which can be expressed as follows:

$$\mathbb{P}\left[b_i = x \mid \{c^{(j)}, d^{(j)}\}_{j=1..N}, \{F_{j,k} > t\}_{j=1..N,k=1..\nu}\right]\,.$$

To obtain this estimator, we will first split the influence of the various error terms $F_{j,k}$ using Bayes' theorem[3]. Then we will derive an expression which can be used to estimate b_i using the value $y = \text{HW}^8_{[kl_c:(k+1)l_c]}(2^i x c^{(j)})$ for each decoding failure. Finally we use experimental measurements to calculate the required probability distributions.

[3] Bayes' theorem: $\mathbb{P}[A|B] = \mathbb{P}[B|A] \cdot \mathbb{P}[A]/\mathbb{P}[B]$

The first step proceeds as follows:

$$\begin{aligned}&\mathbb{P}\left[b_i = x \mid \{c^{(j)}, d^{(j)}\}_{j=1..N}, \{F_{j,k} > t\}_{j=1..N,k=1..\nu}\right]\\ =&\mathbb{P}\left[b_i = x\right] \cdot \frac{P\left[\{F_{j,k} > t\}_{j=1..N,k=1..\nu} \mid b_i = x, \{c^{(j)}, d^{(j)}\}_{j=1..N}\right]}{\mathbb{P}\left[\{F_{j,k} > t\}_{j=1..N,k=1..\nu} \mid \{c^{(j)}, d^{(j)}\}_{j=1..N}\right]}\\ =&\mathbb{P}\left[b_i = x\right] \prod_{j=1}^{N}\prod_{k=1}^{\nu} \frac{\mathbb{P}\left[F_{j,k} > t \mid b_i = x, c^{(j)}, d^{(j)}\right]}{\mathbb{P}\left[F_{j,k} > t\right]}\,.\end{aligned}$$

In the last equation, we assume that individual failures are independent, and that knowledge of $c^{(j)}$ and $d^{(j)}$ without any knowledge of a or b does not help in determining the failure probability of a codeword.

In the second step we use Heuristic 5.2.2, which gives:

$$\mathbb{P}\left[b_i = x\right] \prod_{j=1}^{N} \prod_{k=1}^{\nu} \frac{\mathbb{P}\left[F_{j,k} > t \mid \mathrm{HW}^8_{[kl_c:(k+1)l_c]}(2^i x c^{(j)}) = y\right]}{\mathbb{P}\left[F_{j,k} > t\right]}$$

Looking at the last term, we can use Bayes' again to get the following:

$$\mathbb{P}\left[b_i = x\right] \prod_{j=1}^{N} \prod_{k=1}^{\nu} \frac{\mathbb{P}\left[\mathrm{HW}^8_{[kl_c:(k+1)l_c]}(2^i x c^{(j)}) = y \mid F_{j,k} > t\right]}{P[\mathrm{HW}^8_{[kl_c:(k+1)l_c]}(2^i x c^{(j)}) = y]} \tag{5.2}$$

Both probabilities in the fraction can be estimated for each possible y by generating enough sample ciphertext with the right property and reconstructing the probability distribution experimentally.

A similar derivation can be made for estimating the bits of a_i, by replacing the b and $c^{(j)}$ terms with a and $d^{(j)}$ terms respectively and assuming that knowledge of a_i does not give any practical knowledge of acG. The result is summarized in Corollary 1.

Corollary 1. *Given N failing ciphertexts of Ramstake corresponding to a fixed secret key (a, b) and known values for (c, d), the probability of the bits at position i of a and b can be approximated as:*

$$\begin{aligned}
&\mathbb{P}\left[b_i = x \mid \{c^{(j)}, d^{(j)}\}_{j=1..N}, \{F_{j,k} > t\}_{j=1..N,k=1..\nu}\right] \\
=&\mathbb{P}\left[b_i = x\right] \prod_{j=1}^{N} \prod_{k=1}^{\nu} \frac{\mathbb{P}\left[\mathrm{HW}^8_{[kl_c:(k+1)l_c]}(2^i x c^{(j)}) = y \mid F_{j,k} > t\right]}{P[\mathrm{HW}^8_{[kl_c:(k+1)l_c]}(2^i x c^{(j)}) = y]}
\end{aligned}$$

$$\begin{aligned}
&\mathbb{P}\left[a_i = x \mid \{c^{(j)}, d^{(j)}\}_{j=1..N}, \{F_{j,k} > t\}_{j=1..N,k=1..\nu}\right] \\
=&\mathbb{P}\left[a_i = x\right] \prod_{j=1}^{N} \prod_{k=1}^{\nu} \frac{\mathbb{P}\left[\mathrm{HW}^8_{[kl_c:(k+1)l_c]}(2^i x d^{(j)}) = y \mid F_{j,k} > t\right]}{P[\mathrm{HW}^8_{[kl_c:(k+1)l_c]}(2^i x d^{(j)}) = y]}
\end{aligned}$$

5.3 ATTACK ON RAMSTAKE

With the estimator at hand we can extract the approximate position of the ones in the secret vectors from the probability distribution using statistical methods, and subsequently apply a variant of the Slide-and-Dice attack (called "reduced Slice-and-Dice attack") to find the exact secret vectors. We demonstrate the feasibility of such an extraction by applying a heuristic procedure to a probability distribution resulting from decryption failures in the Ramstake-756839 KEM given in Section 5.1.2.

Assume that we are given N decryption failures and the estimator introduced in the previous section computes the probability that a certain bit position in a, b is zero or one. The computation is following Equation (5.2) in Corollary 1 for different pairs of values $a, d^{(j)}$ and $b, c^{(j)}$. The result are the probabilities $Pr[b_i = 1]$ and $Pr[b_i = 0]$ for each bit position i.

Given these estimates, we can examine the function that maps a bit position b_i to the fraction

$$\varrho_i = \mathbb{P}[b_i = 1]/\mathbb{P}[b_i = 0]\,. \tag{5.3}$$

An example of such a function is shown as a blue line in Figure 5.2, where we denote the (secret) positions of the *ones* the dashed black lines.

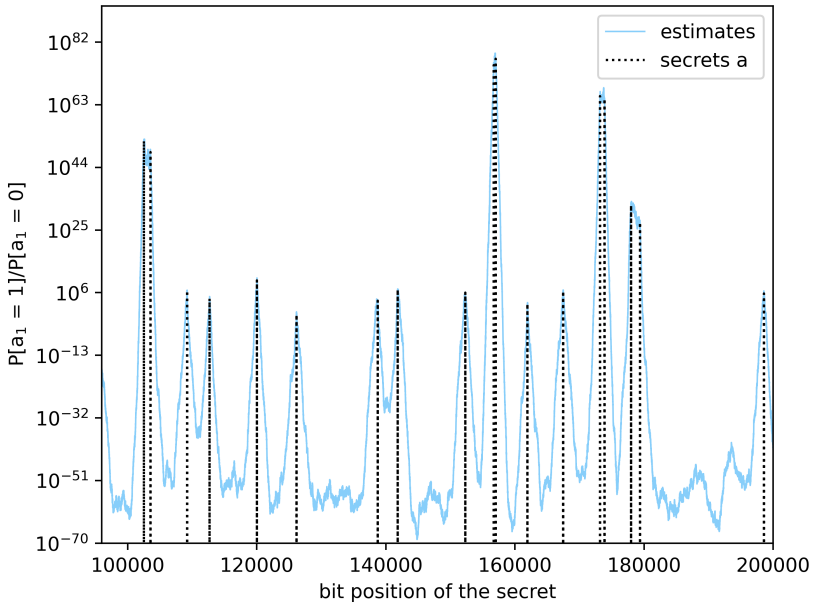


FIGURE 5.2: Function mapping bit positions to the ratio $(\mathbb{P}[b_i = 1]/\mathbb{P}[b_i = 0])$. The blue graph is the output of applying Corollary 1 to $N = 2^{12}$ decryption failures, where the (secret) bit positions of a are marked as dashed, vertical lines.

From Figure 5.2 one can see that the positions of the *ones* correlate to local maxima in the function. However, not all local maxima correspond to a *one*, and there appears to be no clear indication of the distance from a *one* to a local maxima, and neither to the height of the local maxima. Since Corollary 1 only provides us with a probability distribution, we cannot identify bit positions with certainty without further processing.

To apply the Slice-and-Dice attack Section 4.1.3, we need to distinguish intervals that contain a *one* and intervals that contain only *zeroes*. We developed a heuristic procedure, summarized in Figure 5.3, that extracts such intervals for Ramstake.

We provide experiments along the way to show how our heuristic procedure recovers intervals that admit correct parts. At the same time, we will see that some intervals require some random sampling of starting positions to admit a correct partition. In Section 5.3.4 we will conclude, that the experimental estimates are sufficiently good to significantly reduce the number of guesses to construct a partition required to extract the secrets.

5.3.1 *Interval Detection*

In this section we provide details on our methodology for extracting a set of intervals from the probability distribution to sample a *good* partition for the Slice-and-Dice attack. We do not claim that this procedure is optimal, nor are applicable to any Mersenne-based scheme. Our experimental results verify that this procedure provides sufficient information to recover the secret vectors in the Ramstake scheme with *reasonable*[4] effort.

Throughout this section we will consider distinct intervals $\mathtt{I}_{i,l}^{LABEL} = [i, i + l], i, l \in \mathbb{Z}$ with least significant bit i and length l that are either likely to contain a *one* according to the probability distribution given by ϱ_i, labeled

[4]We specify the cost of the attack in Section 5.3.3.

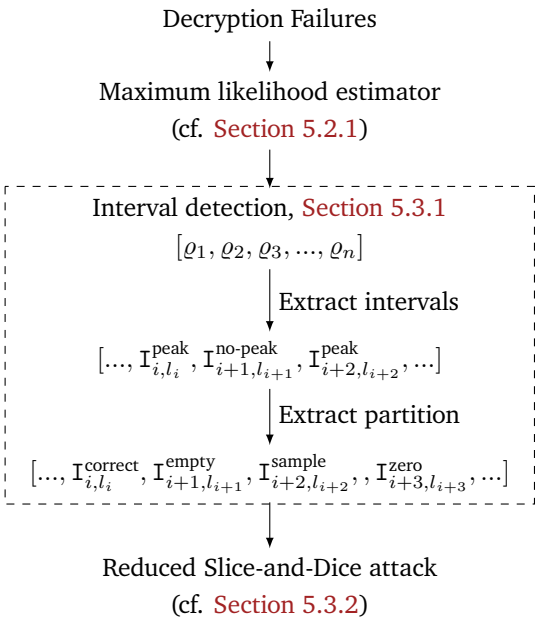


FIGURE 5.3: Overview of the extraction (dashed box) of a partition from the initial probability distribution. Each value ϱ_i correspond to the probability that a bit in a secret vector is either zero, or one as in Equation (5.3). The first step detects intervals $\mathtt{I}^{\text{LABEL}}_{i,l_i}$, LABEL $\in$ {peak, no-peak} where this probability peak (and vice versa). The second step defines a partitioning, where each part $\mathtt{I}^{\text{LABEL}}_{i,l_i}$, LABEL $\in$ {correct, empty, sample, zero} corresponds to a part as used in a reduced variant of Slice-and-Dice attack Figures 5.7 and 5.8.

peak, and intervals that are unlikely to contain a *one*, labeled *no-peak*, Later, we apply a new set of labels corresponding to the intervals suitability to form a part for the Slice-and-Dice attack.

EXTRACT INTERVALS. We apply a heuristic procedure to extract the intervals from the function values of $\varrho_i = (Pr[b_i = 1]/Pr[b_i = 0])$.

1. Threshold segmentation: Let T be the average of all probability ratios ϱ_i. We identify all positions i such that the ratio is larger than the threshold: $\varrho_i \geq T$.

2. Bit ranges "peak": We join all bit position above the threshold T that are at most $n/(10\omega)$ bit[5] positions apart to form a single interval $\mathtt{I}_{i,l}$, beginning at the least and ending at the most significant value above the threshold. These intervals will be labeled as *peak*

3. Bit ranges "no-peak": Each interval between two consecutive "peak" intervals will form an interval $\mathtt{I}^{\textit{no-peak}}_{i,l}$.

4. Width reduction: We reduce the width of all *peak* intervals (thus increasing the size no-peak intervals) relative to the height of their local maxima, i. e., the maxima enclosed by $\mathtt{I}_{i,l}$. Let

$$T_{i,l} = \tau \cdot \sum_{j=i}^{i+l} \varrho_i / l \,,$$

be a local threshold for the interval $\mathtt{I}_{i,l}$. The bounds $i, i+l$ of each bit range are shifted towards the local maxima until the respective

[5] The value $n/(10\omega)$ has been chosen due to the average size n/ω of a balanced partition from our experiments in Section 5.3.

$\varrho_i \geq T_{i,l}$. This step reduces the overall area covered by the bit ranges and gives more accurate intervals around the local maxima. Smaller values may give better partitioning results but increase the risk of excluding a potential position of a one[6].

[6]We chose $\tau = 1/32$ as a heuristic value that worked well in our experimental evaluation.

Applying these steps to all estimates results in segmented intervals of the probability distribution. An example from our experiments is shown in Figure 5.4, where the *peak* intervals are *solid*. At the end of the extraction phase we have a list of intervals, where zero and non-zero intervals are interleaved, for example, a sequence of the form $\mathtt{I}^{peak} = [..., \mathtt{I}^{peak}_{i,l}, \mathtt{I}^{no\text{-}peak}_{i',l'}, \mathtt{I}^{peak}_{i'',l''}, ...]$.

Remark 5. *We note that these observations are tied to our experimental evaluation of the Ramstake cryptosystem in Section 5.3. Further, if a position of a one in the secret is outside of the derived bit range, i. e., as a result of an incorrect estimation, the approximation of the parts and the following application of the reduced Slice-and-Dice attack will not be successful with high probability. However, our empirical results in Section 5.3.4 suggest that the heuristic values have been chosen conservatively enough to circumvent this possibility.*

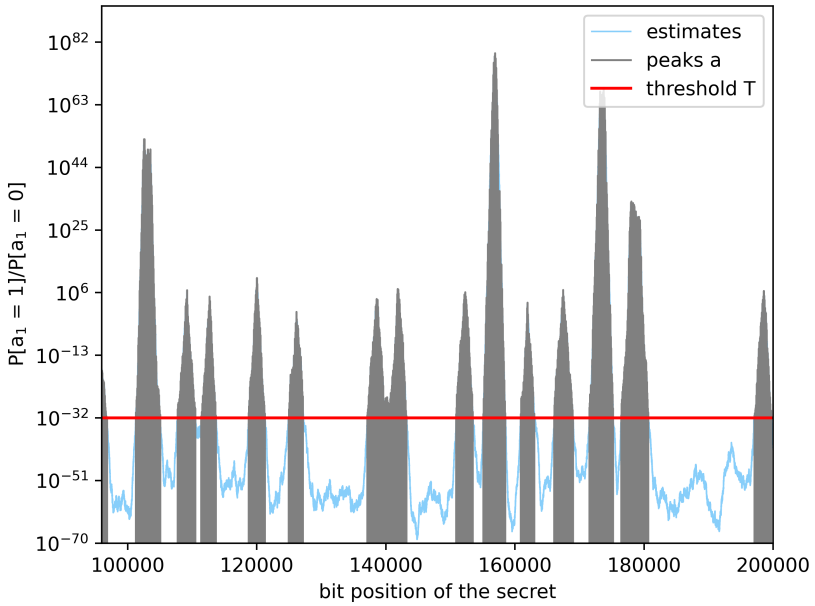


FIGURE 5.4: Identified "peak" (shaded in gray) and "no-peak" (all other) intervals after threshold segmentation, extracting bit ranges and width reduction.

MERGE INTERVALS. Next, we define a new list $\mathtt{I}^{parts}$ of intervals according to the intervals ability to form a correct part in the Slice-and-Dice attack from Section 4.1.3. That means, we find intervals that contain a one only in their lesser significant half. After this procedure the list of intervals contains four possible labels:

- *correct*, if a position of a one is expected in the lower half, corresponding to a correct part.
- *empty*, if it is expected to contain only zeroes, and a *correct* interval is preceeding.

- *sample*, *zero*, where the first are intervals with an expected position of a one that are followed by the latter, which is a zero-interval of shorter length.

Specifically, *correct* and *empty* intervals are identified as follows: Let $\mathtt{I}^{peak}_{i,l_i}, \mathtt{I}^{no\text{-}peak}_{j,l_j}$ be intervals with $i < j$ in the list $\mathtt{I}^{peak}$. If there exists a *no-peak* interval that is larger than at least one previous *peak* interval, i. e., if $l_j > j - i$, we define a new interval $\mathtt{I}^{correct}_{i,2(j-i)}$. The remaining unassigned range is labeled as $\mathtt{I}^{empty}_{l_j-(j-i),j+l_j}$. All intervals in this range are removed from the list $\mathtt{I}^{peak}$. The new intervals are added to the list $\mathtt{I}^{parts}$. An example for this procedure is shown in Figure 5.5.

We further define *sample* and *zero* labeled intervals as follows: Let $\mathtt{I}^{peak}_{i,l_i}$, $\mathtt{I}^{no\text{-}peak}_{j,l_j}$ be two consecutive intervals in the remaining list $\mathtt{I}^{peak}$. For each two consecutive intervals, we have that $l_j < j - 1$, i. e., the intervals containing zeros with high probability are shorter than there respective preceedeing intervals that contains ones with high probability. We relabel all remaining intervals as *peak* $\mapsto$ *sample* and *no-peak* $\mapsto$ *zero* and add them to the list $\mathtt{I}^{part}$. After this procedure we have a list $\mathtt{I}^{part}$. of intervals, where intervals labeled *correct*, *empty*, *sample* and *zero*. We assume that the list is ordered with respect to the least starting position of the intervals.

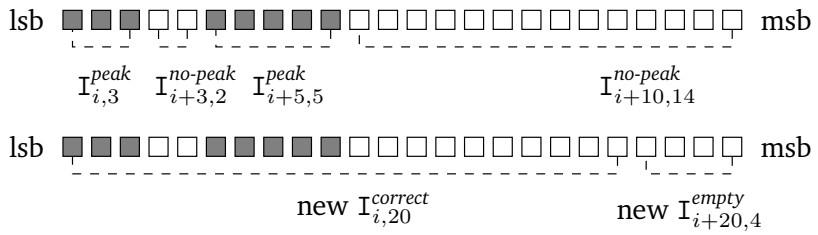


FIGURE 5.5: Merging process for bit ranges. White boxes correspond to a no-peak interval, filled gray boxes to a peak interval.

5.3.2 *Reduced Slice-and-Dice Attack*

We consider a variant of the Slice-and-Dice attack that uses the additional knowledge from the intervals to derive partitions of size for the secrets a, b. The set of all intervals labeled with *correct* already resembles correct parts, where the parts start at the first position of the interval, and thus admit a natural starting position for a part.

The intervals labeled with *sample* are followed by an interval with zero's that is shorter than the *sample* interval, i. e., if one sets the starting position of the part to the first bit in the interval, then the secret bit may not be in the lower half of the part. We solve this by sampling the starting position within the *sample* interval as defined in Figure 5.7. Let $\mathtt{I}_a$ be the intervals for secret a and $\mathtt{I}_b$ the intervals for secret b. Figure 5.6 shows an example of such a partitioning from our implementation of the attack, where the thick, green solid lines correspond to *correct* intervals, and the thin, pink solid lines correspond to the start and end of the *sample* intervals. In Figure 5.8 we give the pseudocode for the reduced Slice-and-Dice attack using the *new* partitioning function.

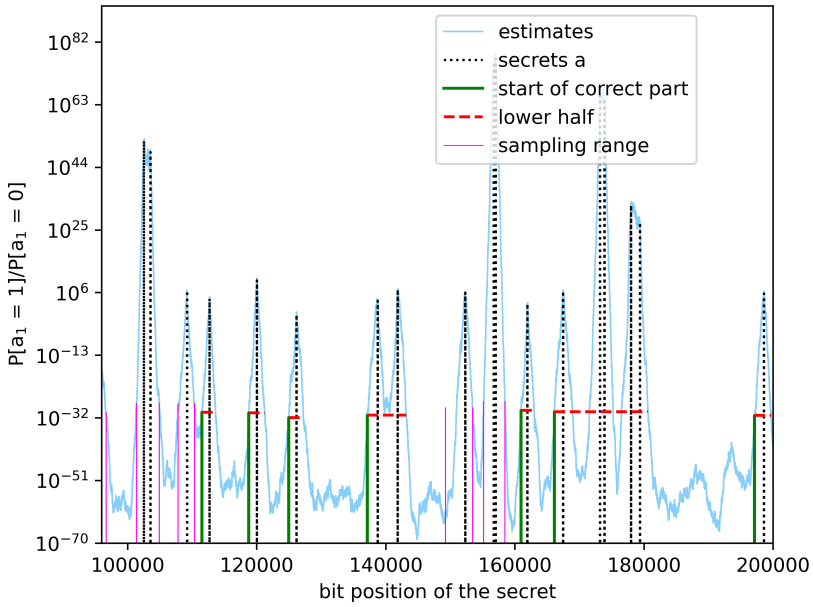


FIGURE 5.6: Partitioning from *correct* intervals enclosed in solid, pink lines and sampling intervals to guess the remaining parts as green lines with dashed red sampling range.

PARTITION($\mathtt{I}$)

$X \leftarrow \{\}$
for $\mathtt{I}_{i,l_{one}}^{\text{LABEL}}$ in $\mathtt{I}$
  if LABEL = *correct*
    $X \leftarrow X \cup i$
  elseif LABEL = *sample*
    Let $\mathtt{I}_{j,l_e}^{empty}$ be the preceeding interval with $j < i$.
    $pos = i - \min(l_e, l_{one}/z)$
    $x \xleftarrow{\$} [pos, i + l_{one}]$
    $X \leftarrow X \cup x$
return X

FIGURE 5.7: Partitioning for Slice-and-Dice attack.

REDUCEDSND($pk, G, p, \mathtt{I}_a, \mathtt{I}_b$)

while True
  $P \leftarrow$ PARTITION($\mathtt{I}_a$)
  $Q \leftarrow$ PARTITION($\mathtt{I}_b$)
  $B \leftarrow$ construct basis for $\mathcal{L}_{a,b,H}$ from P, Q
  $B^* \leftarrow$ LATTICEREDUCTION(B) ▷ Returns short vectors.
  if $\exists b_a^*, b_b^* \in B^*$ s.t. $b_a^* G + b_b^* = pk$
    return b_a^*, b_b^*

FIGURE 5.8: Reduced Slice-and-Dice attack.

In the following examine the success probability that a partition as output by Algorithm 5.7 is correct, particularly, motivate the choice for sampling in the range $[i - \min(l_e, l/z), i + l_{one}]$ for the intervals *sample* intervals. To that end, we give a formula to count the expected number of correct starting positions for a generic sampling range and determine the optimal width for sampling in the empty interval. Finally, we instantiate our result to derive the overall success probability of the attack.

NUMBER OF CORRECT POSITIONS. Consider the bit string spanning an interval $\mathtt{I}^{empty}_{-l_e,l_e}$, an interval $\mathtt{I}^{sample}_{0,l_{one}}$ and an interval $\mathtt{I}^{zero}_{l_{one},l_{zero}}$. Let j be the (unknown) position of a one in this bit string, and let *pos* be the starting position of a part which should serve as input to the Slice-and-Dice attack. Then j is in the lower half of the part defined by *pos* only, if the distance from j to *pos* is at most as large, as the number of zero bits in the remaining part, which ends at position $l_e + l_{one} + l_{zero}$. More formally, this is the case if

$$j - pos \leq l_{one} - j + l_{zero} \,.$$

The number of correct positions can be bounded by

$$min_j := \min(j + l_e, l_{one} - j + l_{zero}) \,,$$

where the first is the number of possible values for $j < pos$, and second is the maximal number of proceeding zeros. That means, larger values of l_e, l_{one} and l_{zero} result in a larger number of correct positions, and the maximal number of useful positions in the *empty* interval can be bounded by the size of the subsequent *sample* and *zero* intervals.

Consider the number of correct position for possible values of j, which are depicted in Figure 5.9.

- The number is maximal, if j is located in the center of the interval $[-l_e, l_{one} + l_{zero}]$, resulting in $(l_e + l_{one} + l_{zero})/2$ correct positions.
- The number is minimal, if j is located on the boundaries of the *sample* interval, i. e., $[0, l_{one}]$.
  - For example, for $j = 0$ the correct starting positions are $[-l_e, 0]$, the cardinality of which is bounded by $\min(l_e, l_{one} + l_{zero}) \leq \min(l_e, 2l_{zero})$.
  - For $j = l_{one} - 1$, a part is correct if it starts in the interval $[l_{one} - l_{zero}, l_{one}]$, resulting in $\min(l_e + l_{one}, l_{zero}) = l_{zero}$ correct positions.

Consequently, the number of correct positions is always bounded by $l_e \leq 2l_{one}$. We bound the position to sample a part with an offset of $\min(l_{one}, l/z)$. In our experimental results in Section 5.3.3 we determine concrete values for z.

AVERAGE NUMBER OF GUESSES. For each individual part we can now express the number of positions that results in a *correct* partition.

$$\mathbb{E}[min_j] = \frac{1}{l_{one}} \sum_{j=1}^{l_{one}} \min(j + l_e, l_{one} - j + l_{zero}) \,,$$

where we assumed, that the secret bit position j is uniformly distributed in $\mathtt{I}^{sample}_{i+l,l_{one}}$, such that each position occurs with probability $1/l_{one}$.

A sampled part is *correct*, if j is in the lower half of the part. We can bound the probability that a part is correct using the value $\min_j$, which is the minimal number of correct positions that may occur for the sample X in Figure 5.7. Let $size(X) = l_{one} - \min(l_e, l_{one}/z)$ be the size of the sampled part

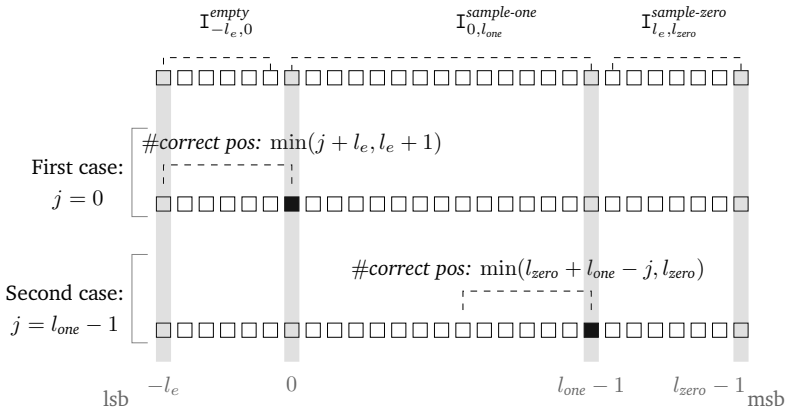


FIGURE 5.9: Number of correct positions for the secret position of the bit if $j = 0$ (First case) and $j = l_{one} - 1$ (Second case) for a successful Slice-and-Dice attack in the intervals $\mathtt{I}^{zero}_{i-1}, \mathtt{I}^{non\text{-}zero}_{i}, \mathtt{I}^{zero}_{i}$.

$$\mathbb{P}\left[j \leq \frac{size(X)}{2}\right] \geq \frac{\min_j}{size(X)},$$

then the expectation is

$$\begin{aligned}\mathbb{E}\left[j \leq \frac{size(X)}{2}\right] \geq& \mathbb{E}\left[\frac{\min_j}{size(X)}\right] \\ =& \frac{\mathbb{E}[\min_j]}{size(X)},\end{aligned}$$

where we assumed that the number of correct positions $\min_j$ and the size of the sampling range are independent, and that j is uniformly distributed over all parts.

Finally, let ω_a, ω_b be the number of ones located in the *sample* intervals. Then the number of guesses until a correct partition is found (and thus number of iterations of the while-loop) in Figure 5.8 is

$$\left(\mathbb{E}\left[j \leq \frac{size(X)}{2}\right]^{\omega_a+\omega_b}\right)^{-1}. \tag{5.4}$$

Remark 6. *The reduced Slice-and-Dice attack can speed-up using Grover's algorithm similar to the* original *Slice-and-Dice attack as described in [TS19]. The corresponding quantum variant would require a number of Grover iterations relative to the square-root of Equation* (5.4).

[TS19] Tiepelt and Szepieniec, "Quantum LLL with an Application to Mersenne Number Cryptosystems"

5.3.3 *Experimental Evaluation*

All phases of the attack can be computed in a few dozens of hours. However, our code was not optimized for speed and large parts can be parallelized. The implementation also generates the diagrams visualizing the attack, of which we present a selection in this work.

The implementation utilizes the Ramstake code submitted to the first round of the NIST post-quantum competition [Nat17]. We modified the error correcting code by reducing the number of codewords to $\nu = 1$ to artificially increase the probability of a decryption failure. The increased failure probability is about 2^{-13} for the simplified Ramstake KEM. This

[Nat17] National Institute for Standards and Technology, *Post-Quantum Cryptography Call for Proposals*

simplification does not, to the best of our knowledge, give any advantage to an adversary except for the *practical* generation of failures.

The demonstrator takes as input the public key $pk := (c, sd)$ and outputs an estimation of the secret key $sk := (a, b)$ as well as the evaluation of the partitioning procedure. To collect failing ciphertexts the demonstrator has access to a decryption oracle in form of the modified Ramstake code that returns a $\top$ in case of a successful key exchange and $\perp$ in case of a decoding error or a re-encryption failure. The latter cases are indistinguishable. Further, our demonstrator does not perform the actual lattice reduction. Instead it takes the secret key a, b as an additional input to evaluate if the approximated intervals allow to derive a *correct* partitioning and computes the respective success probability.

PRECOMPUTATION. First, we generate a set of random decryption failures and successes. The samples allow to estimate the probabilities that a given bit position in a, b causes more than t error bytes in the shared noisy secret S resulting from the encryption with c, d as described in Equation (5.2). This step results in a look-up table mapping bit positions in c, d to failure probabilities for each bit position in the secret. The precomputation can be performed without access to the decryption oracle and only needs to be computed once.

COLLECTING DECRYPTION FAILURES. Next, we collect a set of decryption failures for the attacked secret key by querying the oracle with random ciphertexts and keep those that lead to a decryption failure.

INTERVAL DETECTION. Given the set of decryption failures, one can perform the interval detection as in Section 5.3.1. From our experiments, the *zero* interval has length about $l_{zero} \approx l_{one}/2$ and the *empty* interval have length about $l \approx l_{one}/4$. We examine the success probability that result from our empirical results, the sample range $[i - \min(l_{one}, l_{one}/4), i + l_{one}]$ consists of $((5l_{one})/4)$ bit positions. The number of correct starting positions for j are distributed over the range $[1 : l_{one}]$ as $(l_{one}/4)...(3l_{one}/4)...(7l_{one}/8)...(3l_{one}/4)...(l_{one}/2)$, depicted in Figure 5.10. The probability for a uniformly random part to be *correct* follows as:

$$\mathbb{P}\left[j \leq \frac{size(X)}{2}\right] = \frac{1}{l_{one}} \sum_{j=0}^{l_{one}} \min(j + \frac{l_{one}}{4}, l_{one} - j + \frac{l_{one}}{2}) \approx \frac{39 l_{one}}{80} + \frac{3}{8}\,.$$

And thus the probability for one of the parts the be sampled correctly is at least

$$\frac{(39 l_{one}/80) + (3/8)}{(5 l_{one}/4)} = \frac{39}{80} + \frac{3}{10 l_{one}} \geq \frac{39}{80}\,. \tag{5.5}$$

This results in an overall success probability of about $(39/80)^{\omega_a + \omega_b}$.

5.3.4 *Empirical Cost Results*

We applied our attack to multiple *secrets* generated pseudo-randomly to allow deterministic verification. The precomputation phase has been performed

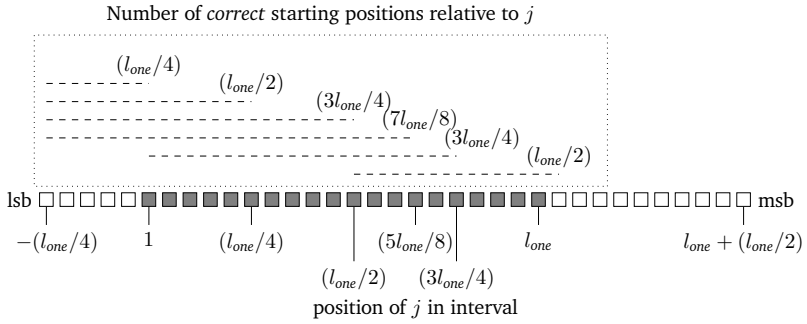


FIGURE 5.10: Distribution of correct positions for $j \in [0 : l_{one} - 1]$ with $l_{zero} = (l_{one}/2)$ and $l = (l_{one}/4)$, where all values have been shifted by 1 to avoid a -1 in the position labels.

TABLE 5.3: Experimental results of our implementation where the column "#ones in *correct*" denotes the number of *secret* positions of ones positioned in *correct* intervals, and "#parts" the number of intervals that require sampling. *#Grover* corresponds to the number of Grover iterations in a quantum variant of the Slice-and-Dice attack to recover the full secret.

Decryption failures	#ones in *correct*	#parts	$\mathbb{P}[success]$ per sampled part	#Grover
2^9	131	125	0.474	2^{68}
2^{10}	161	95	0.477	2^{52}
2^{11}	167	89	0.482	2^{48}
2^{12}	169	88	0.482	2^{46}

using 64 samples generated from the seed "c0ffee". An excerpt of our results is summarized in Table 5.3. An extensive version can be found in Appendix A, showing the average number of positions located in the *correct* intervals, the expected number of parts to be sampled and the expected success probability for each part. Additionally, we give the number of Grover iterations required to recover the secret key (cf. Section 4.1.3).

Throughout our experiments all secret ones could be identified either in *correct* or *sample* intervals. An example of the partitioning is shown in Figure 5.6, where the *correct* parts are solid vertical lines with the lower half is marked as dashed horizontal lines. The sample range is enclosed by the numbered solid vertical lines. The positions of the ones in the secret are added as vertical dashed lines for verification purposes.

COMPLEXITY. The attack requires $2^{64} \cdot (2^{12}/6) \leq 2^{74}$ decryption queries to collect the failing ciphertexts. The factor $1/6$ for the number of decryption failures arises from the existence of 6 failed codewords in the original cipher while our estimates from the simplified implementation use only a single codeword for each failure.

The precomputation and estimation of the secret grows linear in the number of sample and decryption failures and can be neglected. Our estimates allow to capture an average of 168 secret positions in *correct* intervals, suggesting that at most 88 random parts have to be sampled. For the average value of the preceding empty space we have $l_e \approx (l_{one}/4)$, for the trailing empty space $l_{zero} \approx (l_{one}/2)$, thus supporting our analysis. The success

probability for each part follows as 0.484 resulting in about 2^{92} classical guesses or 2^{46} Grover iterations to extract the secret. Figure 5.11 shows the relation between the number of collected failures and the number of Grover iterations required to find the secret. In summary, we can break the IND-CCA security of Ramstake using 2^{74} classical queries and about 2^{46} Grover iterations.

It should be noted that this may be an infeasible number of queries, which is why for evaluation in the NIST process, the attacker is generally constrained to a maximum of 2^{64} queries. Techniques such as failure boosting [DAn+19a], which increase the failure probability of ciphertexts and which have been applied to encryption schemes based on the learning with errors problem, may reduce the number of required decryption queries. Moreover, recent results [DRV19] for these schemes show that information about previous failures can be used to bootstrap the search for new failures. We did not investigate if these techniques are applicable to Mersenne prime schemes.

[DAn+19a] D'Anvers et al., "Decryption Failure Attacks on IND-CCA Secure Lattice-Based Schemes"

[DRV19] D'Anvers, Rossi, and Virdia, *(One) failure is not an option: Bootstrapping the search for failures in lattice-based encryption schemes*

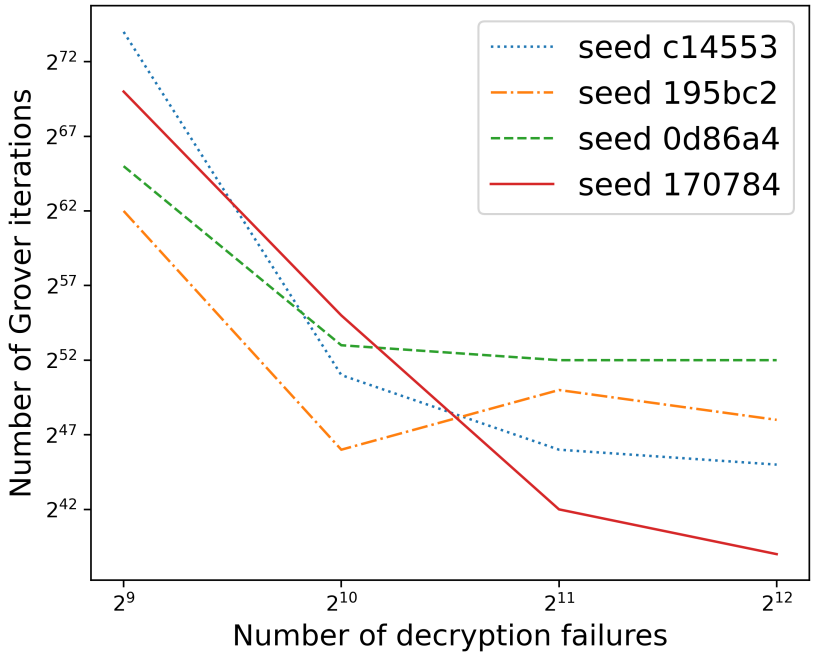


FIGURE 5.11: Outline of number of Grover iterations required for the quantum variant of the reduced Slice-and-Dice attack (cf. Figure 5.8), relative to the number of decryption failures.

IMPLICATIONS ON MERSENNE-756839. The scheme diverges by the error term $\hat{d}$, which introduces $\omega := 128$ additional error positions. We consider those insignificant compared to the large number of about $2\omega^2$ byte errors in each encoding, and therefore assume that a similar heuristic can be constructed given access to decryption failures.

However, the error correcting code deployed by the Mersenne-756839 cryptosystem has a failure probability of 2^{-239}, making it significantly more difficult to find decryption failures. While the precomputation phase can be adapted, the actual attack phase seems infeasible.

6

On the Cost of Universal Signature Forgery in SPHINCS$^+$

Parts of this chapter are verbatim from our publications [BT21a; BT21b].

In this chapter we explore the possibilities of crafting a universal forgery from a second preimage of a hash function in the SPHINCS$^+$ signature scheme. The SPHINCS$^+$ scheme was analyzed during the third round of the NIST competition. As part of the analysis we provide a cost estimation of performing the attack on a fault-tolerant quantum computer.

Objective & Contribution. The SPHINCS$^+$ signature scheme has been chosen by the NIST as one of finalists of the post quantum competition [Nat22]. Consequently, it is important to study how the signature scheme can withstand against from quantum computers, and how the properties on which its security is based on are affected within the scheme. This study explores which points of attack the signature scheme offers, how these points can be best exploited, and what the expected costs are that an adversary would have to implement such an attack on a quantum computer. To the best of our knowledge the specifications from the third round carry over to those of the finalists.

[Nat22] National Institute for Standards and Technology, *NIST: Selected Algorithms 2022*

We are interested in evaluating the security of SPHINCS$^+$ against generic preimage attacks of the underlying hash function, specifically those utilizing QAA. When applying quantum amplitude amplification, the input database is the preimage space of the hash function in question, and the membership oracle compares the hash functions output to a given image. To mount such an attack the hash function has to be implemented as a quantum circuit and used in every iteration of the QAA algorithm (cf. Section 3.1.1). Notably, the computation of the hash function is the dominant cost in each such iteration.

As a first contribution (cf. Section 6.1), we identify the XMSS component in SPHINCS$^+$ to admit the attack with the *cheapest* hash function oracle for Grover's algorithm. Subsequently, in Section 6.4, we estimate the cost of implementing the complete attack in the context of fault-tolerant quantum computers (cf. Section 3.3.3) . As a result, we suggest that a universal forgery can be constructed in $1.55 \cdot 2^{101}$ logical-qubit-cycles, which is defined as the product of surface code cycles and logical qubits (we introduce the metric in Section 6.1), and which is supposed to correspond to the classical hash function invocations. The previously most-well known attack required $2^{129.5}$ [CNS17] classical hash function invocations. We note that this does *not* imply an attack on, or the insecurity of the SPHINCS$^+$ signature scheme.

[CNS17] Chailloux, Naya-Plasencia, and Schrottenloher, "An Efficient Quantum Collision Search Algorithm and Implications on Symmetric Cryptography"

Following the suggestion by NIST to review the security in terms of a maximal depth for quantum circuits, it is clear that even for a depth of 2^{96} the attack cannot be implemented without further constraints. A detailed analysis of the impact if introducing a MaxDepth limitation for the quantum circuits is out of scope for this manuscript.

6.1 On the fault-tolerant cost of computing a second preimage

In the context of fault-tolerant quantum computing, the introduction of error correction mechanisms can result in a significant overhead of resources required to implement an algorithm. This section presents our methodology of estimating the resources across the different layers of a fault-tolerant quantum architecture. including the Application, Logical and Quantum error correction layer (cf. Section 3.3). The objective is to provide an estimation of the attack cost corresponding to resources the quantum error correction layer.

We adopt the concept of logical-qubit-cycles, LQC, as quantum cost metric for our analysis. A logical-qubit-cycle is the product of the surface code cycles and the number of logical qubits to implement the circuit in a fault-tolerant manner. This is motivated by the idea, that the cost of each such cycle is *roughly* equivalent to the cost of a single (classical) hash function invocation [Amy+16, As. 4 and Cost Metric 1], allowing to compare classical and quantum cost. We further consider this metric to be the most fitting in comparison to the time-space product, i. e., number of Grover iterations and number of qubits, for the best generic attack in [CNS17].

[Amy+16] Amy et al., "Estimating the Cost of Generic Quantum Pre-image Attacks on SHA-2 and SHA-3"

[CNS17] Chailloux, Naya-Plasencia, and Schrottenloher, "An Efficient Quantum Collision Search Algorithm and Implications on Symmetric Cryptography"

Let $Q_G^{logical}$ be the number of logical qubits to implement a quantum circuit. Define $Q_{MSD}^{logical}$ to be the number of logical qubits to perform the magic state distillation as outlined in Section 3.3.3. Let $\#\mathrm{SCC}(\textit{SPR-Attack})$ be the number of surface code cycles associated with the implementation of a quantum preimage attack *SPR-Attack*. The number of logical-qubit-cycles is considered to be the total cost of the attack:

$$\mathrm{LQC}(\textit{SPR-Attack}) = \#\mathrm{SCC}(\textit{SPR-Attack}) \cdot (Q_G^{logical} + Q_{MD}^{logical}) \,. \tag{6.1}$$

In the following we provide an overview of the cost elements involved in each layer, and how they carry over to the next lower layer. The estimation is *generous-to-the-attacker*, meaning that we lower bound several cost elements on the way and only consider the *dominant* cost.

6.1.1 *Application Layer*

In Section 6.2, we explore the cost of crafting a universal signature forgery relative to the cost of implementing a membership oracle for Grover's algorithm. We begin by assuming that such an attack requires to compute a second preimage. With this premise in mind, we identify the component within the SPHINCS$^+$ framework that results in the fewest consecutive calls to a hash function. This is done because the computation of the hash functions is the dominant cost to implement an iteration in the search using Grover's algorithm. Additionally, we determine the probability $p_{preimage}$ that

a second preimage attack is successfull, relative to the (pre-)image space of the hash function in question. In the subsequent paragraphs we provide results in support of quantifying the cost of Grover's algorithm in this setting.

Remark 7. *To craft a universal signature sometimes incurs an additional* classical *cost: a forged signature needs to map to the correct address ADRS (cf. Section 4.2.2) in the SPHINCS$^+$ hypertree corresponding to a given public-key secret-key pair. This cost is denoted cost$_{ADRS}$ and addressed individually for each proposed forgery attack. The cost is purely classical, and additive, meaning it does not scale the cost of the second preimage attack, and has no impact on the scaling of the fault tolerant resources. Later, we will show that this additional, classical cost is* much *lower than the quantum cost. To provide a conservative analysis we do not account for this in our final cost metric.*

Success Probability of a Second Preimage Attack. Assume that we are given a hash function $\mathcal{H} : \{0,1\}^{n_1} \to \{0,1\}^{n_2}$ along with a preimage X, and further assume that hash functions perform like random functions. We are interested in the probability $p_{preimage}$ that there exists a second preimage X', such that $\mathcal{H}(X) = \mathcal{H}(X')$, relative to the size of the image and preimage space.

We consider two cases, where the first is equality of the length the preimage and image, $n_1 = n_2$, and the second where the preimage space is larger, i. e., $l \cdot n_1 = n_2$ for $l > 1$. In Section 6.2, we will instantiate the following two lemmas: For the first case, Lemma 2 bounds the probability that such a preimage exists.

Lemma 2. *For a random function $\mathcal{H} : \{0,1\}^{n_1} \to \{0,1\}^{n_2}$, i. e., every image being chosen uniformly at random and independent, with $n_1 = n_2$, given an image y, the probability of a second preimage existing is $p_{preimage} \geq 0.632$.*

Proof.

$$\begin{aligned}
\mathbb{P}\left[\exists x \in \{0,1\}^{n_1} : \mathcal{H}(x) = y\right] &= 1 - \mathbb{P}\left[\forall x \in \{0,1\}^{n_1} : \mathcal{H}(x) \neq y\right] \\
&= 1 - \prod_{x \in \{0,1\}^{n_1}} \mathbb{P}\left[\mathcal{H}(x) \neq y\right] \\
&= 1 - \prod_{x \in \{0,1\}^{n_1}} \frac{2^{n_2} - 1}{2^{n_2}} \\
&= 1 - \left(\frac{2^{n_2} - 1}{2^{n_2}}\right)^{2^{n_1}} \\
&= 1 - \left(1 - \frac{1}{2^{n_2}}\right)^{2^{n_1}} \\
&\geq 1 - \frac{1}{e} \geq 0.632
\end{aligned} \tag{6.2}$$

In the last step we used that $n_1 = n_2$ along with the bound $(1 - 1/y)^y \leq 1/e$. □

For the second case, $l \cdot n_1 = n_2$ with $l > 1$, the probability of a second preimage existing is bound by

$$1 - \left(1 - \frac{1}{2^{n_2}}\right)^{2^{n_2/l}}.$$

Since $(1 - 1/2^{n_2}) < 1$, the success probability of a second preimage existing decreases for increasing values of l. We consider specific values of l and the resulting success probability later.

QUANTUM PREIMAGE SEARCH. Given a string $X \in \mathcal{X}$ a quantum preimage search for a value X' can be implemented using Grover's search algorithm (cf. Section 3.1.1) over the search space $\mathcal{X}$. The membership oracle's implementation is the computation of the function $\mathcal{H}$, the comparison with $\mathcal{H}(X)$, and the uncomputation of $\mathcal{H}$. If the uncomputation is not performed, fresh qubits would be required every iteration, resulting in an exponential increase in memory. The procedure is the *canonical* instantiation of the circuit in Figure 3.3 from Section 3.1.1.

The cost of performing the search has three elements: The cost of implementing a single Grover iterations, i. e., the membership oracle identifying the (second) preimage, the miscellaneous operations such as the inversion over the mean, and the number of repetitions of the Grover iteration.

The cost to implement the miscellaneous operations grows linearly with the search space, but is independent of the remaining attack. The cost to implement the oracle grows with the cost of implementing the membership function in question. In the setting of SPHINCS$^+$, the membership function is the consecutive application of a hash function i. e., $\mathcal{H}(...\mathcal{H}(x))$. The cost of implementing such a function increases with more consecutive invocations, and we denote this number of invocations by $\#\mathcal{H}$. The number of iterations is determined by the ratio of the preimage space and the number of second preimages.

Note that if multiple preimage exist, than Grover's algorithm requires fewer iterations, and may not output a preimage with large probability if the number of iterations is too large. However, even if the number of preimages is not known, the complexity of first determining the number of, and subsequently finding *one* preimage, is still in $O\left(2^{\lambda/2}\right)$, As a simplification and to provide a conservative analysis we assume the number of second preimages to be exactly 1, resulting in $\lfloor \frac{\pi}{4} 2^{\lambda/2} \rfloor$ iterations of the quantum algorithm (cf. Lemma 1).

6.1.2 *Logical Layer*

In Section 6.3, we report on the amount of logical resources required to implement an oracle for Grover' algorithm, the miscellaneous operations, and the total resources to implement the second preimage search. Particularly we report the following values from our Q# implementation:

- The number of CNOT, Clifford and T gates: $G^{\text{CNOT}}, G^{\text{Clifford}}, G^{\text{T}}$
- The T-DEPTH of the quantum circuit: $G^{\text{T-Depth}}$
- The number of logical qubits used: Q^{logical}

6.1.3 *Quantum Error Correction Layer*

We proceed to estimate the amount of resources required to implement the logical resources in a fault-tolerant manner, given faulty gates and qubits with the parameters from Assumption 3.3.1. Our primary resources that we consider are the logical-qubit-cycles LQC, which are the product of surface code cycles #SCC the number of logical qubits $\mathsf{Q}^{\text{logical}}$.

The number of surface code cycles as well as number of logical qubits required implement the quantum circuit in a fault-tolerant manner have two sources: A surface code to embed all CNOT and Clifford gates, and the magic state distillation required to implement T gates. We review our procedure to derive the count of surface code cycles for each component.

SURFACE CODE FOR CNOT AND CLIFFORD GATES. The failure probability of each CNOT and Clifford gate should be at most

$$p_G^{out} \leq \frac{1}{\mathsf{G}^{\text{CNOT}} + \mathsf{G}^{\text{Clifford}}},$$

such that the application of all gates together succeeds with probability

$$\left(1 - \frac{1}{1/p_G^{out}}\right)^{1/p_G^{out}} \leq \frac{1}{e} \in O(1), \tag{6.3}$$

where $1/p_G^{out}$ is the total number of CNOT and Clifford gates. While this would require a constant number of repetitions of the complete algorithm, we assume the success probability to be approximately 1 to provide a more conservative analysis.

With p_G^{out} and the input error probability p_G^{in} of a gate from Assumption 3.3.1 at hand, we can determine the surface code distance d using Assumption 3.3.3, i. e., the smallest d such that Equation (3.1) holds. This determines the number of surface code cycles, i. e., $\#\mathsf{SCC}_G \approx d$, required to implement the CNOT and Clifford gates. Additionally, Assumption 3.3.3 provides the number of physical qubits required to implement the surface code.

MAGIC STATE DISTILLATION FOR T GATES. Similarly, the failure probability of each T gate should be at most

$$p_T^{out} \leq \frac{1}{\mathsf{G}^{\mathsf{T}}},$$

such that the application of all gates succeeds with probability close to 1 (see Equation (6.3)).

One can use the Algorithm of [Amy+16, Alg. 4] and [FDJ13, Sec. 2] to compute the number of layers of magic state distillation required, where Assumption 3.3.3 provides the surface code distance and Definition 3.3.2 provides means to compute the number of surface code cycles required for each distillation. Let $\#\mathsf{SCC}_{MSD}$ be the number of surface code cycles for each layer of T gates.

Note that the T-gates for each gate-layer of the circuit have to be produced to apply the layer, i. e., we can only *apply* a T-gate for which a magic state has been produced. The production of magic states requires formidable amount of logical qubits. However, after a distillation procedure is completed, these

[Amy+16] Amy et al., "Estimating the Cost of Generic Quantum Pre-image Attacks on SHA-2 and SHA-3"

[FDJ13] Fowler, Devitt, and Jones, "Surface code implementation of block code state distillation"

qubits may be reused for future distillation. With Assumption 3.3.2 at hand, we can look at each layer independently. That means, we determine how many parallel magic state distillations should be used to produce the magic states, where a larger number of distilleries increases the overall number of logical qubits required. Specifically, given a number of magic states #MagicStatesPerDistillery produced by each distillery, one can identify the number #Distilleries of Distilleries required such that

$$\frac{\mathsf{G}^{\mathsf{T}}}{\mathsf{G}^{\text{T-Depth}}} > \#\text{MagicStatesPerDistillery} \cdot \#\text{Distilleries}\,.$$

Logical Qubit Cycles. To compute the number of logical-qubit-cycles we consider the dominant number of surface code cycles, i. e., the number of surface code cycles that dominates the fault-tolerant implementation. For each layer of T gates, $\#\mathsf{SCC}_{MSD}$ surface code cycles are required.

With Assumption 3.3.2, the CNOT and Clifford gates are spread evenly over all layers and all qubits in the quantum circuit, the average number of surface code cycles required on each such layer is

$$\mathsf{G}^{G}/(\mathsf{G}^{\text{T-Depth}} \cdot \mathsf{Q}^{\text{logical}}) \cdot \#\mathsf{SCC}_{G}\,,$$

where $G \in \{\text{CNOT}, \text{Clifford}\}$. The number of surface code cycles used to compute the logical qubit cycles is then

$$\#\mathsf{SCC} = \max(\mathsf{G}^{G}/(\mathsf{G}^{\text{T-Depth}} \cdot \mathsf{Q}^{\text{logical}}) \cdot \#\mathsf{SCC}_{G}, \#\mathsf{SCC}_{MSD})\,.$$

6.2 UNIVERSAL SIGNATURE FORGERIES IN SPHINCS$^+$

A universal forgery attack enables the construction of a signature on an arbitrary message, such that the signature is verified correctly when presented with a challenge public key. In this context, the procedures in the subsequent subsections are from the perspective of an attacker who is provided with a public key and access to a signing oracle that provides valid signatures. From this perspective, we consider the cost relative to breaking the preimage resistance of a hash function using quantum amplitude amplification. Consequently, we identify the attack that minimizes the cost to implement the membership oracle. All of the proposed attacks follow the same structure, and the resulting forged signature is composed of the three components from Table 6.1.

Table 6.1: Notation for attack procedures on SPHINCS$^+$.

Encoding	Description
X	Part of a valid signature from the signing oracle.
$\overline{Y}$	Part of a signature from a *freshly* generated public-key secret-key pair.
$\hat{Z}$	Part of a universal forgery that connects X and $\overline{Y}$ using a second preimage.

Assume in the following that we want to produce a valid signature $\hat{\sigma}^{\overline{M}}_{\text{SPHINCS}^+}$ on message $\overline{M}$. We apply the procedure in Figure 6.1 to the

four components, the initial message digest $\mathcal{H}$, the FORS signature, the WOTS signatures and the XMSS signatures. For each component we analyze the cost on the Application layer.

Step 1 Request a signature

$$\sigma^{M}_{\text{SPHINCS}^+} = (R, \sigma^{\overline{vk}_{\text{FORS}}}_{HT}, \sigma^{\overline{md}}_{\text{FORS}}),$$

signing a message M.

Step 2 Generate a *fresh* SPHINCS$^+$ public-key secret-key pair $\overline{vk}_{\text{SPHINCS}^+}$, $\overline{sk}_{\text{SPHINCS}^+}$ that can be used to sign any message $\overline{M}$. Let $\overline{M}$ be the target message for the universal forgery.

Step 3 Generate a fresh signature

$$\overline{\sigma}^{\overline{M}}_{\text{SPHINCS}^+} \leftarrow \textsc{Sign}(\overline{sk}_{\text{SPHINCS}^+}, \overline{M}).$$

Step 3a For some attacks, Step 3 has to be repeated multiple times to find a randomness $\overline{R}$ that results in the *correct* parameter $\hat{ADRS}$. This incurs an additional cost of cost_{ADRS}.

Step 4 Compute a second preimage $\hat{X}$ in a hash function in one of SPHINCS$^+$ components.

Step 5 Embed the preimage $\hat{X}$ and the genuine signature $\sigma^{M}_{\text{SPHINCS}^+}$ into the freshly generated signature $\overline{\sigma}^{\overline{M}}_{\text{SPHINCS}^+}$ of the target message, resulting in the forgery $\hat{\sigma}^{\overline{M}}_{\text{SPHINCS}^+}$.

FIGURE 6.1: Attack procedure for a preimage attack on SPHINCS$^+$.

6.2.1 *(Universal) Forgery from Pre-Image in the Message Digest*

Recall that the initial message digest in the signature generation of the SPHINCS$^+$ scheme as in Section 4.2.2 computes the map

$$md, id_{tree}, id_{leaf} \leftarrow \mathcal{H}(R, vk_{\text{SPHINCS}^+}.sd, vk_{\text{SPHINCS}^+}.root, M),$$

where $|md|$ depends on the height of the hypertree and the FORS instance[1], $M \in \{0,1\}^*$ is the arbitrarily sized message, $R \in \{0,1\}^\lambda$ is a random string and $vk_{\text{SPHINCS}^+}.sd, vk_{\text{SPHINCS}^+}.root$ are fully defined by the corresponding public key.

[1]The value l is called m in the SPHINCS$^+$ specification [Hül+20].

FORGERY. The universal forgery attack on the message digest is a special case, where the second and third step of procedure in Figure 6.1 can be skipped. This is the case, because a colliding message digest immediately results in a forged signature. Note that the attack requires to compute a second preimage rather than *any* collision, because one of the preimages is already fixed by the requested signature from Step 1.

Since the goal is to forge a signature on *any* message, the only degree of freedom to manipulate is the random string R. Therefore, one needs to find a preimage $\hat{R}$, such that

$$md, id_{tree}, id_{leaf} \leftarrow \mathcal{H}_{msg}(\hat{R}, vk_{\text{SPHINCS}^+}.sd, vk_{\text{SPHINCS}^+}.root, \overline{M}) ,$$

which can be achieved by running QAA with $\hat{R} \in \{0,1\}^\lambda$ as the input search space, performing Step 4 in Figure 6.1. The forged signature is then

$$\hat{\sigma}^{\overline{M}}_{\text{SPHINCS}^+} = (\hat{R}, \sigma_{HT}, \sigma^{\overline{M}}_{\text{FORS}}) .$$

Figure 6.2 shows the placement of the attack in the SPHINCS$^+$ framework.

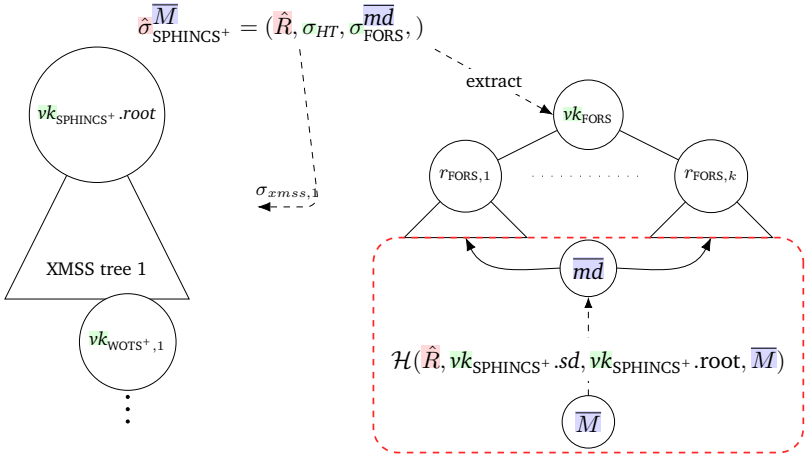


FIGURE 6.2: Simplified SPHINCS$^+$ scheme with the universal forgery using a second preimage of the message digest for message $\overline{M}$ with preimage $\hat{R}$ in the dashed box.

COST. The forgery requires a preimage search with a single invocation of the hash function, thus $\#\mathcal{H} = 1$. Forging a signature using the initial message digest does not require any overhead from finding a the correct address *ADRS*, i. e., Step 3a in Figure 6.1 does not need to be applied, since the hypertree of the honest signature σ remains the same. Correspondingly, $\text{cost}_{ADRS} = 0$.

SUCCESS PROBABILITY. The bit length l_{md} of the primary message digest[2] used in the signature of the subsequent component (the FORS instance), is of fixed length [Hül+20, Sec. 6.1]

$$\begin{aligned} &|md| + |id_{tree}| + |id_{leaf}| \\ &= \left\lfloor \frac{(k \log t + 7)}{8} \right\rfloor + \left\lfloor \frac{(h - \frac{h}{d} + 7)}{8} \right\rfloor + \left\lfloor \frac{\frac{h}{d} + 7}{8} \right\rfloor > \lambda , \end{aligned} \tag{6.4}$$

depending on the number of layers of the hypertree, the total height of the hypertree and the number of leaves and trees in the FORS instance.

With the random bit string $\hat{R} \in \{0,1\}^\lambda$ being shorter than the message digest, we can compute the probability of a preimage existing as in Table 6.2. The table shows that for all parameter sets, the probability is much smaller than 1, but strictly larger than 2^{-88}.

[2] The primary message digest is called "tmp_md" in the specification [Hül+20].

[Hül+20] Hülsing et al., *SPHINCS+-Submission to the 3rd round of the NIST post-quantum project*

TABLE 6.2: Probability that a second preimage exists (cf. Section 6.1.1, Equation (6.2)) when searching over preimage space of the randomness R of bitlength λ and the image space of the message digest *md* of length according to Equation (6.4). The ratio $l > 1$ corresponds to the relative size of the exponents of the pre-/image space. The table shows the probability for all SPHINCS$^+$ parameter sets (cf. Table 4.1).

λ	$\lvert md\rvert$	$\geq l$	$\leq \mathbb{P}\left[\exists x \in \{0,1\}^\lambda : \mathcal{H}(x) = y\right]$
128	184	1.44	2^{-56}
128	214	1.67	2^{-86}
192	254	1.32	2^{-62}
192	280	1.45	2^{-88}
256	323	1.26	2^{-67}
256	331	1.29	2^{-75}

Corollary 2 (Message Digest Forgery)**.** *There exists an algorithm that computes a universal signature forgery with probability at least* 2^{-88}*, and* $cost_{ADRS} = 0$*, that requires to compute a second preimage of the hash function* $\mathcal{H}$ *with quantum oracle depth at most* $\#\mathcal{H} = 1$*.*

Remark 8. *The message digest attack gives rise to an existential forgery attack with a single oracle depth, and a lower failure probability of* $p_{preimage} \geq 0.632$ *i. e., namely when including the message space of* $\overline{M}$ *for the second preimage, since this allows to match the value of* n_1 *to that of* n_2*. As such the two inputs that can be modified and exchanged in the preimage attack are* M *and* R *However, note that the cost of perform the second preimage search also scales with the size of the search space, which may result in a more expensive attack.*

6.2.2 *Universal Forgery from a FORS Signature*

Recall the FORS signature as described in Section 4.2.2. A signature

$$\sigma^{md}_{\text{FORS}} = \{sk^{(i,j)}_{\text{FORS}}, Auth_{\text{FORS},i}\}_i$$

consists of a FORS secret key (a leaf of the tree) and an authentication path from the leaf to the root, which is the FORS public key. Each node of the hash tree is the output of a hash function $\mathcal{H}_2$ that takes as input the two parent nodes, each of bit-length λ, and outputs a $n = \lambda$ bit hash value. All hash trees are combined by computing the hash of all $r_{\text{FORS},i}$ nodes, resulting in the a virtual root node vk_{FORS} that combines the FORS instance in a single node.

FORGERY. We follow the steps from Figure 6.1, where in Step 3 the FORS signature is computed as

$$\overline{\sigma}^{\overline{md}}_{\text{FORS}} = \{\overline{sk}_i, \overline{Auth}_i\}_i \,.$$

During a verification, the FORS public key is computed as

$$\overline{vk}^{\overline{md}}_{\text{FORS}} \leftarrow \mathcal{H}(vk_{\text{SPHINCS}^+.sd}, \overline{ADRS}, \overline{r}_{\text{FORS},1}, \overline{r}_{\text{FORS},2}, \ldots, \overline{r}_{\text{FORS},k}) \,.$$

As a first step, the signature generation step needs to be repeated multiple times as in Step 3a of Figure 6.1, to find a string $\hat{R}$ that results in an address

$\hat{ADRS}$ that maps to the same FORS instance as $ADRS$. We will analyze the cost of this later.

To forge a signature, one can compute a second preimage for the FORS public key on one of the authentication paths, where $\overline{r}_{\text{FORS},1}$ is the hash of two parent nodes p_1, p_2, i. e.,

$$\overline{r}_{\text{FORS},1} \leftarrow \mathcal{H}_2(vk_{\text{SPHINCS}^+}.sd, \hat{ADRS}, \overline{p}_1, \overline{p}_2)\,.$$

Note that one of the two parents, w.l.g. say p_2, is fixed, for example by depending on the hash with the public key seed as input, and can not be manipulated in order to provide a universal forgery.

That means, the function in question for the second preimage is

$$\overline{vk}^{\overline{md}}_{\text{FORS}} \leftarrow \mathcal{H}_k(vk_{\text{SPHINCS}^+}.sd, \hat{ADRS}, \mathcal{H}_2(\overline{vk}.sd, \hat{ADRS}, \hat{p}_1, \overline{p}_2), \overline{r}_{\text{FORS},2}, \ldots, \overline{r}_{\text{FORS},k})\,.$$

The corresponding QAA uses $\hat{p}_1 \in \{0,1\}^n$ as the input search space, performing Step 4 from Figure 6.1. The forged signature is then

$$\hat{\sigma}^{\overline{M}}_{\text{SPHINCS}^+} = (\hat{R}, \sigma^{vk_{\text{FORS}}}_{HT}, \hat{\sigma}^{\overline{M}}_{\text{FORS}})\,.$$

Figure 6.3 shows the placement of the attack in the SPHINCS$^+$ frameworks, Figure 6.4 the particular attack on the FORS scheme.

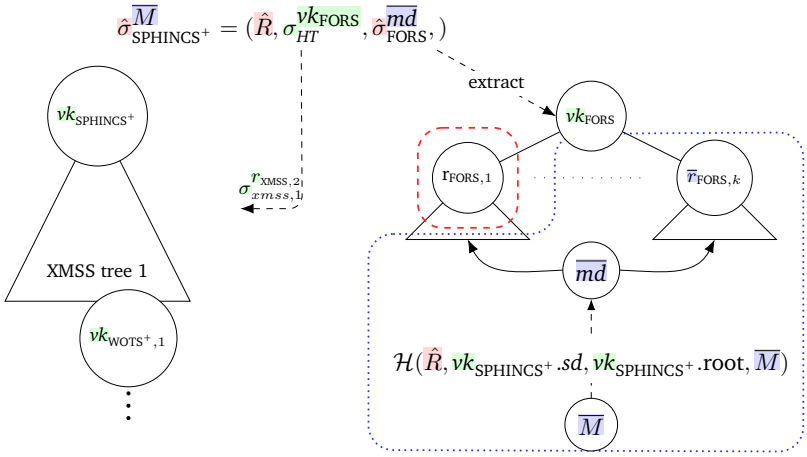


FIGURE 6.3: Simplified SPHINCS$^+$ scheme with the target for the seconds preimage attack for the universal forgery for message $\overline{M}$ in the dashed red box. The dotted blue box marks the *fresh* signature $\overline{\sigma}$.

COST. The only degree of freedom are the parents in the authentication path $Auth_{\text{FORS},i}$. As a result, the second preimage has to be computed from the first level of any of the FORS tree, and consequently requires to chain two iterations of $\mathcal{H}_2$, namely one for the first layer of the FORS tree, and one to compute the $\overline{vk}_{\text{FORS}}$ from all the FORS root nodes. As such, we can implement a forgery attack with a second preimage search of oracle depth $\#\mathcal{H} = 2$.

Next, we analyze the cost of Step 3a (c.f. Section 6.2), the repeated generation of a signature to match the *correct* FORS instance. When construction a SPHINCS$^+$ signature, we can choose the FORS address resulting in the WOTS$^+$ instance associated with the genuine signature σ. Recall

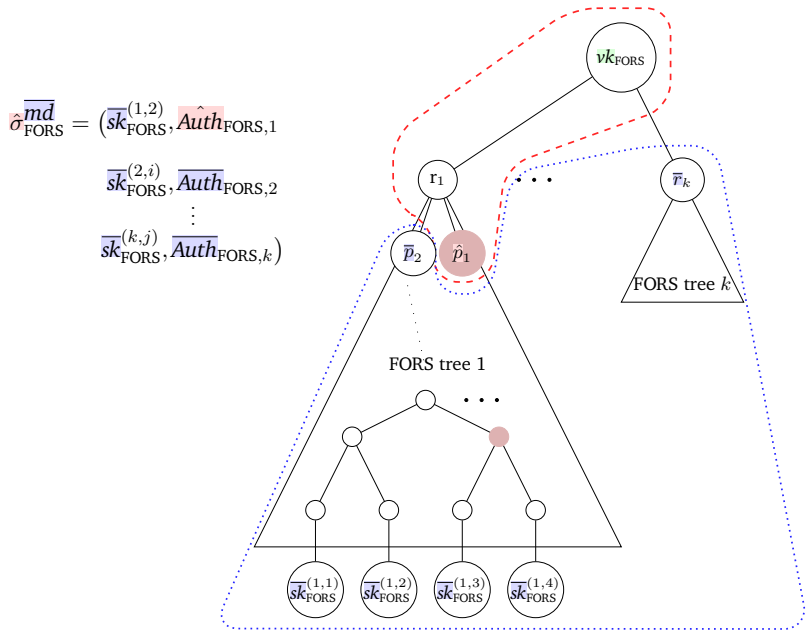


FIGURE 6.4: Simplified SPHINCS$^+$ scheme with the attacked part for the universal forgery for message $\overline{M}$ in the dashed red box. The position of the generated FORS signature in the SPHINCS$^+$ tree and of the forged values in the respective FORS tree are encapsulated in the dotted blue box. Note the position of the leaf depends in the message, and the string of that leaf is determined by public components, thus the roots that are hashed into the vk_{FORS} cannot be chosen freely. The first node that can be used for a preimage attack the sibling p_1 on a second level of a hash tree.

that the SPHINCS$^+$ hypertree has d layers, where the first layer has a single XMSS tree with the SPHINCS$^+$ public-key root. This XMSS tree has $\frac{h}{d}$ leaves, each connected to an XMSS tree. Accordingly, layer $d-1$ has $2^{h-h'}$ XMSS trees, and a total of $2^{h-h'} \cdot 2^{h'} = 2^h$ WOTS$^+$ public keys. That means that each SPHINCS$^+$ hypertree corresponds to 2^h FORS instances. The probability of hitting the correct instance when generating a new signature is $\text{cost}_{ADRS} = 2^{-h}$. Note that for all parameters in SPHINCS$^+$ $h \leq 68$ [Hül+20, Table 3].

[Hül+20] Hülsing et al., *SPHINCS+- Submission to the 3rd round of the NIST post-quantum project*

SUCCESS PROBABILITY. When attacking the FORS component the input and output space of the second preimage search have the same dimension, thus we can apply Lemma 2 to find a preimage with failure probability at most $p_{preimage} \geq 0.632$ as in Lemma 2.

Corollary 3 (FORS Forgery)**.** *There exists an algorithm that computes a universal signature forgery with probability at least* $1 - p_{preimage} \geq 0.632$*, and* $\text{cost}_{ADRS} \leq 2^{-68}$*, that requires to compute a second preimage of the hash function* $\mathcal{H}_2$ *with quantum oracle depth at most* $\#\mathcal{H} = 2$*.*

6.2.3 *Universal Forgery from WOTS$^+$ Signature*

Recall the WOTS signature as described in Section 4.2.2. A signature $\sigma_{\text{WOTS}^+,1}^{\cdot}$ signs either a block of a FORS public key, or a block of any of the root nodes of the XMSS trees. Without loss of generality say that we forge a signature on a node $r_{\text{XMSS},2}$. Note that this is without loss of generality since the bitlength of all nodes and the FORS public key, as well as the procedure

to produce the signature are the same. For the sake of readability, we denote the concatenation of $r_{\text{XMSS},2}$ and its checksum as $X := r_{\text{XMSS},2}||\text{check}_{r_{\text{XMSS},2}}$ as in Section 4.2.2.

The signature is a chain of hash function invocations evaluated on a WOTS$^+$ secret key $sk_{\text{WOTS}^+,i}$, namely the mapping

$$\sigma^{X_i}_{\text{WOTS}^+,i} \leftarrow \mathcal{H}_1(...(\mathcal{H}_1(sk_{\text{WOTS}^+,i}))) ,$$

where the number of hash function invocations depends on the integer representation of the block of length w. Input and output bitlength of the hash function are n bits each.

FORGERY. Follow the procedure in Figure 6.1, where in Step 3 the WOTS$^+$ signature for the i^{th} block is computed as

$$\overline{\sigma}^{\overline{X}}_{\text{WOTS}^+} = \mathcal{H}_1(...\mathcal{H}_1(vk_{\text{WOTS}^+.sd}, \overline{ADRS}, \overline{sk}_{\text{WOTS}^+,i})) , \tag{6.5}$$

where the hash chain has been iterated $\left(\overline{X}_i\right)_w$ times, with $\overline{X}_i$ the i^{th} message block. This step needs to be repeated multiple times (see Step 3a of Section 6.2) to find a randomness $\hat{R}$ mapping to address $\hat{ADRS}$ that maps to the same WOTS$^+$ instance as $ADRS$. We will analyze the cost of this later.

During a verification, the WOTS$^+$ public key is computed by completing the $w - \left(\overline{X}_i\right)_w$ iterations of the hash chain, and hashing all public key blocks, i. e.,

$$vk_{\text{WOTS}^+} \leftarrow \mathcal{H}_l(vk_{\text{WOTS}^+,1}, vk_{\text{WOTS}^+,2}, ..., vk_{\text{WOTS}^+,l}) .$$

l is the number of blocks and each $vk_{\text{WOTS}^+,i}$ is computed similar to Equation (6.5). To forge a signature, one can compute a second preimage for the WOTS$^+$ public key on one of public keys $vk_{\text{WOTS}^+,i}$.

$$vk_{\text{WOTS}^+} \leftarrow \mathcal{H}_l(\mathcal{H}_1(...\mathcal{H}_1(vk_{\text{SPHINCS}^+}.sd, \hat{ADRS}, \hat{\sigma}_{\text{WOTS}^+,i})), \overline{vk}_{\text{WOTS}^+,2}, \overline{vk}_{\text{WOTS}^+,l}) .$$

The resulting QAA uses $\sigma_{\text{WOTS}^+,i} \in \{0,1\}^n$ as the input search space, performing Step 4 in Section 6.2. The forged signature is

$$\hat{\sigma}^{\overline{M}}_{\text{SPHINCS}^+} = (\hat{R}, \hat{\sigma}^{\overline{vk_{\text{FORS}}}}_{HT}, \overline{\sigma}^{\overline{M}}_{\text{FORS}}) .$$

Figure 6.5 shows the placement of the attack in the SPHINCS$^+$ framework, and Figure 6.6 the forgery on the WOTS$^+$ signature.

COST. The input to the hash function, i. e., $\overline{sk}_{\text{WOTS}^+,i}$ is itself an image of a hash function computation. That means, the value cannot be chosen freely during signature generation, but depends on the forgery message md. As such, for a universal forgery, there is no control over the number of chained hash function invocations.

In [Hül+20, Table 3], the block-size is defined as $w = 16$, such that the number of hash function invocations is in the interval $[1, 15]$. If $\mathcal{H}_1$ acts like a random function, then one can compute the expected number[3] X_w of hash function invocations, i. e.,

$$\mathbb{E}[X_{16}] = \sum_{i=1}^{15} i \cdot \frac{1}{15} = 8$$

Consequently, a forgery attack with a second preimage search can be implemented with oracle depth $1 \leq \#\mathcal{H} \leq 15$, with an expected value of 8 for an undetermined message.

[Hül+20] Hülsing et al., *SPHINCS+-Submission to the 3rd round of the NIST post-quantum project*

[3] One could alternatively consider the trade-off of a depth 1 with probability $\geq \frac{1}{15}$, or depth ≤ 15 with probability 1.

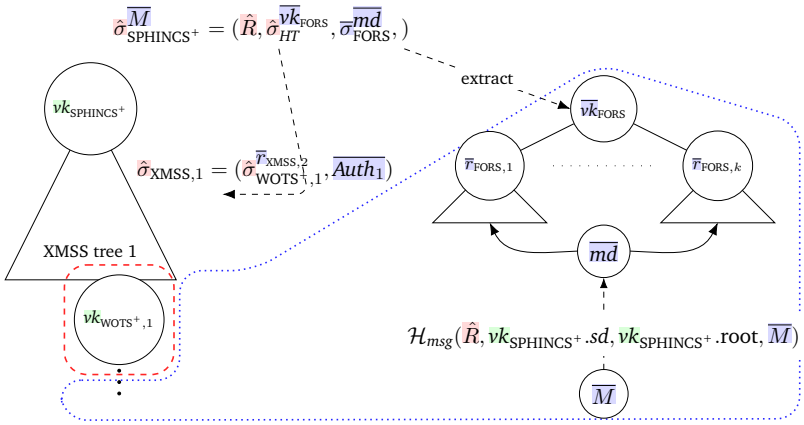


FIGURE 6.5: Simplified SPHINCS$^+$ scheme with the target for the seconds preimage attack for the universal forgery for message $\hat{M}$ in the dashed red box. The dotted blue box marks the *fresh* signature $\overline{\sigma}$.

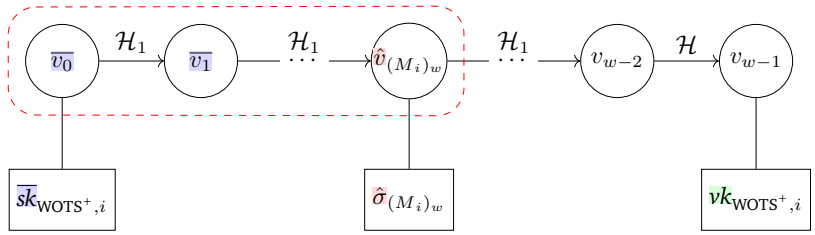


FIGURE 6.6: Simplified SPHINCS$^+$ scheme with the attacked part for the universal forgery for message $\hat{M}$ in the dashed red box.

Once again, when creating a SPHINCS$^+$ signature, we can choose the FORS address which will give us some WOTS$^+$ instance, and we have to pick the *correct* address to match the genuine WOTS$^+$ instance of σ. To forge a WOTS$^+$ instance on the first layer of the hypertree, consider only layer 1 with $2^{h'} = 2^{h/d}$ WOTS$^+$ instances. The probability that a random FORS instance uses the correct WOTS$^+$ instance is $2^{-h/d}$, i. e., $\text{cost}_{ADRS} = 2^{-h/d}$. For [Hül+20, Table 3], this cost is $\text{cost}_{ADRS} \geq 2^{-9}$ for all parameter choices.

[Hül+20] Hülsing et al., *SPHINCS+-Submission to the 3rd round of the NIST post-quantum project*

SUCCESS PROBABILITY. When attacking the WOTS$^+$ component the input and output space of the second-preimage search have the same dimension, thus we can apply Lemma 2 to find a preimage with success probability $1 - p_{preimage} \geq 0.63$ as in Lemma 2.

Corollary 4 (WOTS$^+$Forgery). *There exists an algorithm that computes a universal signature forgery with probability at least* $1 - p_{preimage} \geq 0.63$*, and* $cost_{ADRS} \leq 2^{-9}$*, that requires to compute a second preimage of the hash function* $\mathcal{H}_1$ *with quantum oracle depth at most* $1 \leq \#\mathcal{H} \leq 15$.

6.2.4 *Universal Forgery from a XMSS Authentication Path*

Recall the XMSS signature as described in Section 4.2.2. A signature $\sigma^{r_{\text{XMSS},i+1}}_{\text{XMSS},i} = (\sigma^{r_{\text{XMSS},i+1}}_{\text{WOTS}^+,i,j}, Auth_{\text{XMSS},i})$ consists of a WOTS$^+$ signature of the root of the previous XMSS tree and an authentication path within the XMSS tree. While we already covered the possibility of forging a WOTS$^+$ signature, we review the possibility of attacking the XMSS authentication path. In such an authentication path, each node consists of the hash of its parents.

The root node, which is the SPHINCS$^+$ public key for the topmost tree, is the hash of a node that descended from a WOTS public key (and from the signed message and randomness), and from another node, which can be freely chosen. Particularly, input and output bitlength of the hash function are n bits each.

FORGERY. Following the steps in Figure 6.1, where in the third step the XMSS signature of layer 1 is

$$\overline{\sigma}^{\overline{r}_{\text{XMSS},1}}_{\text{XMSS}} = \overline{\sigma}^{\overline{r}_{\text{XMSS},2}}_{\text{WOTS}^+,1}, \overline{Auth}_{\text{XMSS},1} ,$$

During verification, the XMSS root is computed as

$$vk_{\text{SPHINCS}^+}.root \leftarrow \mathcal{H}_2(vk_{\text{SPHINCS}^+}.sd, \overline{ADRS}, \overline{p}_1, \overline{p}_2) ,$$

where p_1, p_2 are the parents of the root node.

To forge a signature, one can compute a second preimage for the root public key on one of the authentication paths,

$$vk_{\text{SPHINCS}^+}.root \leftarrow \mathcal{H}_2(vk_{\text{SPHINCS}^+}.sd, \overline{ADRS}, \hat{p}_1, \overline{p}_2) .$$

One of the two parents, w.l.g. say p_2, is determined by the honest and fake signature instance, and can not be manipulated in order to provide a universal forgery. The preimage can be retrieved running QAA with $\hat{p}_1 \in \{0,1\}^n$ as the input search space, performing Step 4 of Figure 6.1. The forged signature in Section 6.2 is then

$$\hat{\sigma}^{\overline{M}}_{\text{XMSS},1} = (\overline{R}, \hat{\sigma}^{\overline{vk}_{\text{FORS}}}_{HT}, \overline{\sigma}^{\overline{M}}_{\text{FORS}}) .$$

Figure 6.7 shows the placement of the attack in the SPHINCS$^+$ framework, and Figure 6.8 in the XMSS tree.

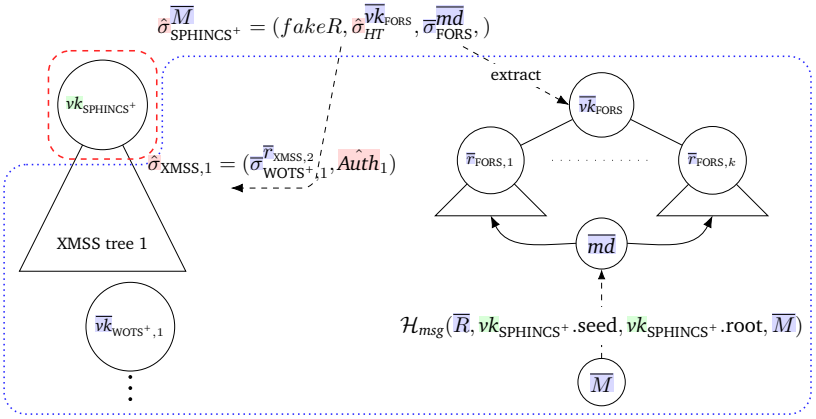


FIGURE 6.7: Simplified SPHINCS$^+$ scheme with the target for the second preimage attack for the universal forgery for message $\overline{m}$ in the dashed red box. The dotted blue box marks the *fresh* signature $\hat{\sigma}$.

COST. Since the input value, i. e., the sibling node, has the same bit-length as the output, i. e., the public key, the probability of finding a preimage is equal to Lemma 2. Also, since the public key is the direct hash of the two parents, the depth of the oracle for an arbitrary message is always $\#\mathcal{H} = 1$.

No address computation is required, as both the WOTS$^+$ and FORS instances checked during verification are generated from the freshly generated public-key pair.

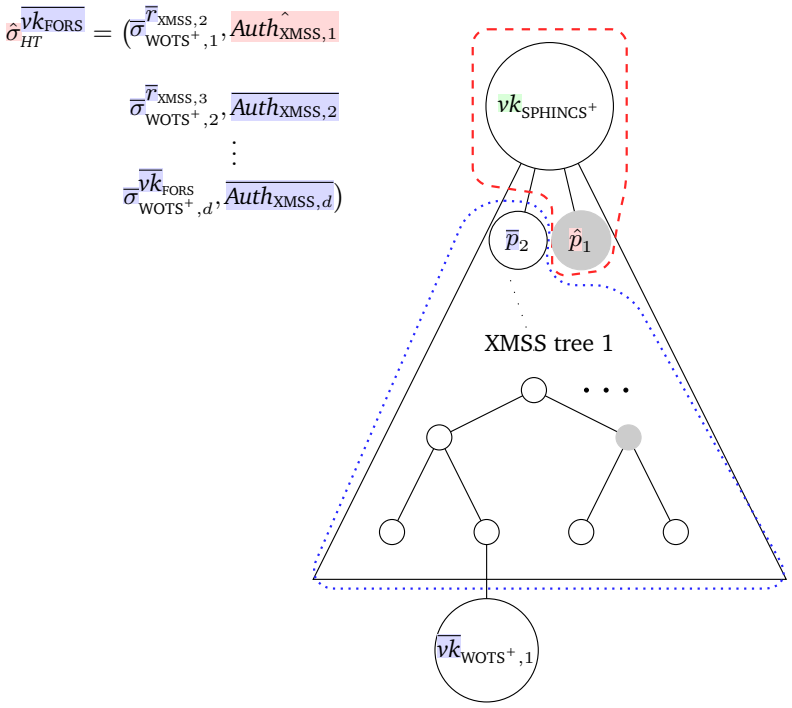


FIGURE 6.8: Simplified SPHINCS$^+$ scheme with the attacked part for the universal forgery for message $\hat{M}$ in the dashed red box and the fake signature in the dotted blue box.

SUCCESS PROBABILITY. When attacking the XMSS component the input and output space of the second-preimage search have the same dimension, thus we can apply Lemma 2 to find a preimage with probability $1-p_{preimage} \geq 0.63$ as in Lemma 2.

Corollary 5 (XMSS Forgery). *There exists an algorithm that computes a universal signature forgery with probability at least* $1 - p_{preimage} \geq 0.63$*, and* $cost_{ADRS} = 1$*, that requires to compute a second preimage of the hash function* H *with quantum oracle depth at most* $\#\mathcal{H} = 1$.

6.2.5 *Comparison of Attack Strategies*

When comparing the four possible places to attack the SPHINCS$^+$ scheme as in Table 6.3, one can see that a second preimage provides a for a universal forgery for each one of the individual components. However, only for the XMSS and the message digest the forgery can be implemented with an oracle depth of one, i. e., the oracle for QAA consists of a single iteration of the keyed hash function. Further, the success probability is maximal for the XMSS component.

As such, we consider this the *cheapest* attack relative to the QAA oracle. In the remainder of this chapter, we will focus on estimating the cost for this individual attack strategy.

TABLE 6.3: Overview of resources in the Application layer to forge a universal signature in SPHINCS$^+$.

Target	Component	$\#\mathcal{H}$	$1 - p_{preimage}$	cost_{ADRS}
$\mathcal{H}(M, R, ...)$	Message Digest	1	$\geq 2^{-88}$	0
$\sigma^{md}_{\text{FORS}}$	FORS	2	≥ 0.63	$\geq 2^{-68}$
$\sigma^{vk_{\text{FORS}}}_{HT}$	WOTS$^+$	$1 \leq i \leq 15$	≥ 0.63	$\geq 2^{-9}$
$Auth_{\text{XMSS}}$	XMSS Path	1	≥ 0.63	0

6.3 QUANTUM CIRCUIT GATE COST

We report estimation results on the logical layer by implementing the membership oracle of the second preimage attack for Grover's algorithm in Q#. The oracle corresponds to the procedure to attack the XMSS component, since this results in the Grover oracle with the fewest number of consecutive hash function evaluations, and thus the *cheapest* oracle. The Q# implementation implementation includes SHAKE-256 and Haraka, both of which are used to instantiate the hash function in SPHINCS$^+$. A detailed description can be found in [BT21a, Sec. 3]. We report the cost of:

[BT21a] Berger and Tiepelt, "On Forging SPHINCS$^+$-Haraka Signatures on a Fault-Tolerant Quantum Computer"

- The Haraka and SHAKE-256 function in Q# in Table 6.4.
- The membership oracle using the quantum circuits for Haraka and SHAKE-256 in Table 6.5.

TABLE 6.4: Resource requirements for a Grover oracle for a preimage attack on SHAKE-256 and the Haraka-based sponge hash function with different input and output lengths [BT21a, Table 3].

Hash function	λ	G^T	G^{CNOT}	$G^{Clifford}$	T-DEPTH	$Q^{logical}$
Haraka	128	$1.16 \cdot 2^{20}$	$1.32 \cdot 2^{21}$	$1.45 \cdot 2^{18}$	$1.06 \cdot 2^{17}$	$1.37 \cdot 2^{10}$
SHAKE-256	128	$1.13 \cdot 2^{20}$	$1.21 \cdot 2^{22}$	$1.29 \cdot 2^{18}$	$1.78 \cdot 2^{11}$	$1.69 \cdot 2^{11}$
Haraka	256	$1.16 \cdot 2^{21}$	$1.32 \cdot 2^{21}$	$1.45 \cdot 2^{19}$	$1.06 \cdot 2^{18}$	$1.62 \cdot 2^{10}$
SHAKE-256	256	$1.13 \cdot 2^{20}$	$1.21 \cdot 2^{22}$	$1.29 \cdot 2^{18}$	$1.17 \cdot 2^{12}$	$1.81 \cdot 2^{11}$
Grover Diffusion						
	128	$1.73 \cdot 2^{10}$	$1.24 \cdot 2^{11}$	2^{10}	$1.11 \cdot 2^{10}$	–
	256	$1.74 \cdot 2^{11}$	$1.24 \cdot 2^{12}$	2^{11}	$1.12 \cdot 2^{11}$	–

Next, we consider the cost to implement Grover's search algorithm. With the number of Grover iterations as $\lfloor \frac{\pi}{4} 2^{\lambda/2} \rfloor$, a second preimage attack on a hash function with search space $\{0,1\}^\lambda$ for $\lambda = 128$ requires $1.57 \cdot 2^{63}$ iterations, for $\lambda = 256$ one requires $1.57 \cdot 2^{127}$ iterations.

Table 6.6 shows the estimation for the cost of a preimage attack on Haraka and SHAKE-256, which are the entries of Table 6.5 scaled with the number of Grover iterations. The logical cost will used as a baseline in the following Section 6.4 to estimate the fault-tolerant cost to mount an attack.

TABLE 6.5: Gate count for our implementation of the Grover components in one Grover iteration.

SPHINCS$^+$*	G^T	G^{CNOT}	$G^{Clifford}$	T-DEPTH	$Q^{logical}$
*-128-Haraka	$1.2 \cdot 2^{21}$	$1.3 \cdot 2^{22}$	$1.4 \cdot 2^{19}$	$1.1 \cdot 2^{18}$	$1.4 \cdot 2^{10}$
*-128-SHAKE-256	$1.1 \cdot 2^{20}$	$1.2 \cdot 2^{22}$	$1.3 \cdot 2^{18}$	$1.8 \cdot 2^{11}$	$1.7 \cdot 2^{11}$
*-256-Haraka	$1.2 \cdot 2^{21}$	$1.3 \cdot 2^{22}$	$1.4 \cdot 2^{19}$	$1.1 \cdot 2^{18}$	$1.6 \cdot 2^{10}$
*-256-SHAKE-256	$1.7 \cdot 2^{20}$	$1.2 \cdot 2^{22}$	$1.3 \cdot 2^{18}$	$1.2 \cdot 2^{12}$	$1.8 \cdot 2^{11}$
Grover Diffusion					
128 bit	$1.7 \cdot 2^{10}$	$1.2 \cdot 2^{11}$	2^{10}	$1.1 \cdot 2^{10}$	–
256 bit	$1.7 \cdot 2^{11}$	$1.2 \cdot 2^{12}$	2^{10}	$1.1 \cdot 2^{11}$	–

TABLE 6.6: Resource estimate for a preimage search to forge an XMSS signature from [BT21a, Table 4], i. e., Table 6.5 scales with the number of Grover iterations. The number of qubits, $Q^{logical}$, is consistent for all Grover iterations (cf. Table 6.5).

SPHINCS$^+$*	GCOST	G^T	G^{CNOT}	$G^{Clifford}$	T-DEPTH
*-128-Haraka	$1.68 \cdot 2^{86}$	$1.89 \cdot 2^{84}$	$1.02 \cdot 2^{86}$	$1.10 \cdot 2^{83}$	$1.73 \cdot 2^{81}$
*-128-SHAKE-256	$1.20 \cdot 2^{86}$	$1.73 \cdot 2^{83}$	$1.88 \cdot 2^{85}$	$1.02 \cdot 2^{82}$	$1.84 \cdot 2^{75}$
*-256-Haraka	$1.69 \cdot 2^{151}$	$1.89 \cdot 2^{149}$	$1.02 \cdot 2^{151}$	$1.11 \cdot 2^{151}$	$1.74 \cdot 2^{145}$
*-256-SHAKE-256	$1.34 \cdot 2^{151}$	$1.89 \cdot 2^{149}$	$1.88 \cdot 2^{150}$	$1.04 \cdot 2^{147}$	$1.37 \cdot 2^{141}$

6.4 FAULT-TOLERANT RESOURCE ESTIMATION

In this section, we give cost estimates of carrying out the *most promising* attack on SPHINCS$^+$ signatures on the quantum error correction layer. We consider the fault-tolerant resources required to mount the attack, specifically, to run the Grover algorithm with the resources reported in Section 6.3 on a physical quantum computer. In particular, we analyze the resource requirements for the SPHINCS$^+$-128 parameter sets, i. e., Haraka and SHAKE-256 hash function. The analysis follows the approach in [Amy+16], where we optimize the parallelization for the magic state distillation.

[Amy+16] Amy et al., "Estimating the Cost of Generic Quantum Pre-image Attacks on SHA-2 and SHA-3"

6.4.1 *Cost Estimation using Haraka*

We follow the procedure from Section 6.1.3 to calculate the dominant number of surface code cycles to embed the CNOT and Clifford gates, or to produce magic states. From Section 6.3, the counts in question are the number of T gates $G^T_{\text{Haraka}} = 1.89 \cdot 2^{84}$, the number of CNOT and Clifford gates $G^{\text{CNOT}}_{\text{Haraka}} = 1.02 \cdot 2^{86}$, $G^C_{\text{Haraka}} = 1.1 \cdot 2^{83}$, the T-DEPTH $G^{\text{T-depth}}_{\text{Haraka}} = 1.73 \cdot 2^{81}$ and the number of qubits $Q^{\text{logical}}_{\text{Haraka}} = 1.1 \cdot 2^{10}$.

SURFACE CODE FOR CNOT AND CLIFFORD GATES. We first consider the surface code as outlined in Section 3.3.3: The target output error probability is $p_G^{out} = 1/\left(G^{\text{CNOT}}_{\text{Haraka}} + G^C_{\text{Haraka}}\right)$. With $p_G^{in} = 10^{-5}$, the smallest surface code distance that fulfills this is $d = 16$. Accordingly, the number of surface code cycles are $\#\text{SCC}_{G,\text{Haraka}} = 16$

Each of the $Q^{\text{logical}}_{\text{Haraka}}$ logical qubits requires 578 physical qubits. In total, the algorithm requires $Q^{\text{physical}}_{\text{Haraka}} \approx 1.853 \cdot 2^{19}$ physical qubits.

MAGIC STATE DISTILLATION. Next, we consider the cost of the magic state distillation as in Section 3.3.3. Given the desired output error rate relative to the size of the circuit

$$p_{MSD}^{out} = 1/\mathrm{G}_{\text{Haraka}}^{\text{T-depth}} ,$$

and the Assumption 3.3.3 given in Section 3.3.3 one can determine the number of layers of magic distillation required.

We use the algorithms [Amy+16, Alg. 4] and [FDJ13, Sec. 2], which take as inputs the assumptions on physical error gate rate (cf. Assumption 3.3.1) and assumptions on the surface code (cf. Assumption 3.3.3), as well as the logical cost (reported in Section 6.3), and outputs the number of distillation layers, the surface code distance d_i for each layer, as well as the number of magic states produced in each distillery. Our implementation of the algorithms can be used to reproduce the numbers.

[Amy+16] Amy et al., "Estimating the Cost of Generic Quantum Pre-image Attacks on SHA-2 and SHA-3"

[FDJ13] Fowler, Devitt, and Jones, "Surface code implementation of block code state distillation"

As a result, two layers of distillation are required, the first with a distance of $d_1 = 19$, the second with a distance of $d_2 = 9$, where $i = 1$ is the top layer, i. e., that outputs the final states. Table 6.7 shows the parameters and the used qubits. Each distillery produces 4 magic states, and requires $\mathrm{Q}_{\text{MD, Haraka}}^{\log} = 240$ logical qubits and $\#\mathrm{SCC}_{\text{Haraka}}^{MSD} = 460$ surface code cycles.

We review the individual layers of T gates. On each layer, an average of

$$\frac{\mathrm{G}_{\text{Haraka}}^{\mathrm{T}}}{\mathrm{G}_{\text{Haraka}}^{\text{T-depth}}} \leq 9 ,$$

T-gates are applied. Therefore, we need about three distilleries to produce a sufficient amount of magic states in each layer.

The bottom layer, $i = 2$ uses more qubits than the top layer, but runs in fewer cycles, we propose to parallelize all three distilleries, computing the bottom layers consecutively, allowing to compute all top layers in parallel at the same time on the qubits used for the bottom layer as in Figure 6.9. In total, the number of logical qubits for the parallelized distillery is $\mathrm{Q}_{\text{MD, Haraka}}^{\log} = 240$.

TABLE 6.7: Magic state distillation scheme for attacking SPHINCS$^+$-128. Distance of the error correcting code d_i and number of logical and physical qubits for each layer i.

Layer	i	d_i	$\mathrm{Q}_i^{\log}$	$\mathrm{Q}_i^{\text{phy}}$
Top	1	19	16	$1.56 \cdot 2^{13}$
Bottom	2	9	240	$1.46 \cdot 2^{15}$

RESULTS. We determine the dominant number of surface code cycles: The average number of gates per layer of T-depth for each CNOT

$$\frac{\mathrm{G}_{\text{Haraka}}^{\text{CNOT}}}{\mathrm{Q}_{\text{Haraka}}^{\text{logical}} \cdot \mathrm{G}_{\text{Haraka}}^{\text{T-depth}}} \cdot \#\mathrm{SCC}_{G,\text{Haraka}} \approx 0.0143 \cdot 16 = 2.288 ,$$

and each Clifford gate

$$\frac{\mathrm{G}_{\text{Haraka}}^{C}}{\mathrm{Q}_{\text{Haraka}}^{\text{logical}} \cdot \mathrm{G}_{\text{Haraka}}^{\text{T-depth}}} \cdot \#\mathrm{SCC}_{G,\text{Haraka}} \approx 0.002 \cdot 16 = 0.32 ,$$

is significantly smaller than the number of surface code cycles required to implement a single layer required for magic state distillation.

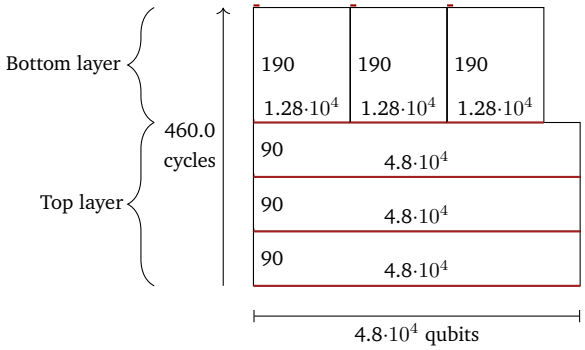


FIGURE 6.9: Magic state distillation scheme for attacking SPHINCS$^+$-128. Pipelining the production of 3 magic states allows to reuse the qubits from the bottom layer in the top layer.

Therefore, the total number of surface code cycles for the entire algorithm is dominated by the magic state distilleries, which is

$$\#\text{SCC} = 460 \cdot \text{G}^{\text{T-depth}}_{\text{Haraka}} \approx 1.5 \cdot 2^{90} .$$

This results in the total cost of the attack (according to Equation (6.1)) of

$$\text{LQC}_{\text{Haraka}}(\textit{SPR-Attack}) \approx 1.54 \cdot 2^{101} .$$

6.4.2 *Cost Estimation using SHAKE-256*

We repeat the procedure from Section 6.1.3 to calculate the dominant number of surface code cycles when instantiating the hash function with SHAKE-256. Following Section 6.3, the number of T-gates are $\text{G}^{T}_{\text{SHAKE-256}} = 1.2 \cdot 2^{86}$, the number of CNOT and Clifford gates are $\text{G}^{\text{CNOT}}_{\text{SHAKE-256}} = 1.88 \cdot 2^{86}$, $\text{G}^{C}_{\text{SHAKE-256}} = 1.88 \cdot 2^{85}$, the T-DEPTH is $\text{G}^{\text{T-depth}}_{\text{SHAKE-256}} = 1.84 \cdot 2^{75}$ and the number of logical qubits are $\text{Q}^{\text{logical}}_{\text{SHAKE-256}} = 1.1 \cdot 2^{10}$.

SURFACE CODE FOR CNOT AND CLIFFORD GATES. The surface code's output error probability is $p^{out}_{G} = 1/\left(\text{G}^{\text{CNOT}}_{\text{SHAKE-256}} + \text{G}^{C}_{\text{SHAKE-256}}\right)$. With $p^{in}_{G} = 10^{-5}$, the smallest surface code distance that fulfills this is $d = 25$; the number of surface code cycles is $\#\text{SCC}_{G,\text{SHAKE-256}} = 25$.

Consequently, each of the $\text{Q}^{\text{logical}}_{\text{SHAKE-256}}$ logical qubits requires 1953 physical qubits with a total of $\text{Q}^{\text{physical}}_{\text{SHAKE-256}} \approx 1.61 \cdot 2^{22}$ qubits.

MAGIC STATE DISTILLATION. With the algorithms [Amy+16, Alg. 4] and [FDJ13, Sec. 2] we get the same number of layers and thus the values for magic state distillation are reminiscent to those of Section 6.4.1, in particular, of Table 6.7. $\text{Q}^{\text{logical}}_{\text{MSD, SHAKE-256}} = 240$. $\#\text{SCC}^{MSD}_{\text{SHAKE-256}} = 460$ surface code cycles. Reviewing the individual layers of T gates shows that on each layer an average of

$$\frac{\text{G}^{\text{T}}_{\text{SHAKE-256}}}{\text{G}^{\text{T-depth}}_{\text{SHAKE-256}}} \leq 249 ,$$

T-gates are applied. Therefore, we need about 83 distilleries to produce a sufficient amount of magic states in each layer.

[Amy+16] Amy et al., "Estimating the Cost of Generic Quantum Pre-image Attacks on SHA-2 and SHA-3"

[FDJ13] Fowler, Devitt, and Jones, "Surface code implementation of block code state distillation"

RESULTS. Comparing the dominant number of surface code cycles, i. e., for CNOT gates

$$\frac{\mathrm{G}^{\text{CNOT}}_{\text{SHAKE-256}}}{\mathrm{Q}^{\text{logical}}_{\text{SHAKE-256}} \cdot \mathrm{G}^{\text{T-depth}}_{\text{SHAKE-256}}} \cdot \#\mathrm{SCC}_{G,\text{SHAKE-256}} \approx 0.31 \cdot 25 = 7.75\,,$$

and Clifford gates

$$\frac{\mathrm{G}^{C}_{\text{SHAKE-256}}}{\mathrm{Q}^{\text{logical}}_{\text{SHAKE-256}} \cdot \mathrm{G}^{\text{T-depth}}_{\text{SHAKE-256}}} \cdot \#\mathrm{SCC}_{G,\text{SHAKE-256}} \approx 0.021 \cdot 25 = 0.525\,,$$

shows that both are smaller than the number of surface code cycles for magic state distillation.

The total number of surface code cycles from magic state distilleries is

$$\#\mathrm{SCC} = \approx 1.6 \cdot 2^{84}\,.$$

The total number of logical qubits required is $1.33 \cdot 2^{14}$. The total cost of running the attack is (cf. Equation (6.1))

$$\mathrm{LQC}_{\text{SHAKE-256}}(\textit{SPR-Attack}) \approx 1.17 \cdot 2^{99}\,.$$

6.4.3 *Comparing Costs*

We compare the logical-qubit-cycles to the number of classical hash function invocations from [Ber+11] required to perform the same attack, since we consider that the former corresponds to the latter as in [Amy+16, As. 4 and Cost Metric 1]. Table 6.8 suggest that the number of logical-qubit-cycles required to implement the proposed universal forgery (cf. Section 6.2) are fewer than the number of classical invocations of the Haraka hash function. We want to highlight that this only affects the Haraka hash functions, and only SPHINCS$^+$-128, and holds conditional on the assumptions on the quantum computing technology made.

[Ber+11] Bertoni et al., *Cryptographic sponge functions*

[Amy+16] Amy et al., "Estimating the Cost of Generic Quantum Pre-image Attacks on SHA-2 and SHA-3"

Otherwise, we compare the number of logical-qubit-cycles to the number of Grover iterations and number of qubits required for generic attacks as reported by [CNS17, Sec. 1.1], neglecting the exponential classical memory cost. We consider the "time-space" product, i. e., the number of Grover iterations multiplied with the number of logical qubits. This time-space product considers resources required on a logical layer, while the logical-qubit-cycles considers resources on the fault-tolerant layer. We chose this comparison, because we are not aware if any previous work on quantifying the cost of a universal forgery with the Haraka hash function, and both metrics consider time and space resources in the quantum setting. A comparison of all results, for both SHAKE-256 and Haraka, with additional information can be found in Table 6.8.

[CNS17] Chailloux, Naya-Plasencia, and Schrottenloher, "An Efficient Quantum Collision Search Algorithm and Implications on Symmetric Cryptography"

A generic collision attack from [CNS17] attacks the internal state of the hash function and requires $\tilde{O}(2^{2n/5})$ Grover iterations, $O(n)$ qubits and $\tilde{O}(2^{n/5})$ classical memory, where $n = 256$ is the size of the internal state. Similarly, a generic preimage attack [CNS17] requires $\tilde{O}(2^{3n/7})$ Grover iterations, $O(n)$ qubits and $\tilde{O}(2^{n/7})$ classical memory. The comparison in Table 6.8 shows that the fault-tolerant cost may even be lower than the logical cost of a generic attack for the Haraka hash function. This is surprising,

since the latter does not consider the additional cost incurred through error correction. However, this may be a result from the fact that our attacks target a search space relative to the security parameter rather than to the internal state of the hash function.

TABLE 6.8: Fault-tolerant cost for our XMSS-attack from Section 6.2.4 using the SHAKE-256 and Haraka hash functions. The number of Grover iterations and time-space product of the generic collision and second preimage attack refers to cost of [CNS17], where $n = 256$ corresponds the size of internal state of Haraka, which is relevant for the generic attack. The rows of our primary comparison are highlighted.

			SHAKE-256	Haraka
SPHINCS$^+$-128	#Classical hash function invocations		–	$2^{129.5}$
Our Attack on SPHINCS$^+$-128	#Distilleries	ϕ	83×3	3×3
	#Log. Qubits	Q^{log}	$1.33 \cdot 2^{14}$	$1.03 \cdot 2^{11}$
	#Total Phys. Qubits	Q^{phy}	$1.03 \cdot 2^{23}$	$1.94 \cdot 2^{20}$
	#Total ECC cycles	#SCC	$1.6 \cdot 2^{84}$	$1.5 \cdot 2^{90}$
	logical-qubit-cycles	LQC	$1.17 \cdot 2^{99}$	$1.55 \cdot 2^{101}$
Our Attack on SPHINCS$^+$-256	#Distilleries	ϕ	42×4	9×1
	#Log. Qubits	Q^{log}	$1.3 \cdot 2^{17}$	$1.16 \cdot 2^{15}$
	#Total Phys. Qubits	Q^{phy}	$1.73 \cdot 2^{25}$	$1.79 \cdot 2^{23}$
	#Total ECC cycles	#SCC	$1.02 \cdot 2^{152}$	$3.95 \cdot 2^{154}$
	logical-qubit-cycles	LQC	$1.31 \cdot 2^{169}$	$1.44 \cdot 2^{171}$
Comparison with logical resources of generic attacks				
Collision Attack on SPHINCS$^+$-128	#Grover Iterations		–	$1.32 \cdot 2^{102}$
	time-space product		–	$1.31 \cdot 2^{110}$
Preimage Attack on SPHINCS$^+$-128	#Grover Iterations		–	$1.64 \cdot 2^{109}$
	time-space product		–	$1.64 \cdot 2^{117}$

Remark 9. *Following Assumption 3.3.1 each surface code cycles requires* $200ns$*. This suggests that running all cycles corresponding to the preimage search for Haraka would require* $1.17 \cdot 10^{13}$ *years. The same algorithm for SHAKE-256 would run for* $2.02 \cdot 10^{11}$ *years.*

7

The Cost of Quantum Lattice Enumeration

Parts of this chapter are taken verbatim from our publications [Bin+24; Bin+23].

Cryptographic constructions based on the hardness of computational problems over algebraic lattices have achieved significant popularity in recent years. Part of the reason for this popularity is the conjectured security of protocols built on them against quantum adversaries, due to the apparent resistance of lattice problems against quantum attacks.

LWE [Reg05; Reg09] is a popular hardness assumption for constructing such post-quantum secure primitives, such as Kyber [Sch+22]. One of the main approaches to solving LWE is via block lattice reduction algorithms, such as BKZ [SE94; SE94]. After building a *lattice embedding* of dimension n of a given LWE challenge, block reduction algorithms will call $O(\text{poly}(n))$ times an SVP solver, such as lattice enumeration, in dimension $\beta < n$. This process results in a better basis for the LWE lattice embedding that allows the attacker to recover the LWE challenge secret. In our work, we consider modern versions of BKZ using early-termination and approximate-SVP solvers, which have been shown [LN20; Alb+21] to be effective while requiring to find a "short enough" lattice vector, rather than the shortest. In order to go from a LWE challenge to an embedding and a choice of BKZ block size β, we use the `lwe-estimator` [APS15b].

The leading cost of block lattice reduction (and therefore, often, of the attacks overall) comes from solving instances of the (approximate [LN20; Alb+21]) shortest vector problem in high dimension. The leading choice for (approximate) SVP solvers are lattice point enumeration [Kan83; FP85; GNR10; CN11; ANS18; Alb+20a] and sieving [AKS01; NV08; Laa15; Bec+16; Alb+19a] algorithms. Due to the central role these algorithms play in the cryptanalysis of lattice-based constructions [LP11; Alk+16; Alb+18] and because multiple post-quantum soon-to-be standards are lattice-based [Sch+22; Lyu+22; Pre+22], clearly understanding their cost is crucial.

Several *quantum* speedups on sieving have been proposed [LMv13; Kir+19; CL21; Bon+23]. These algorithms improve upon their classical counterparts using a quantum search or a quantum walk to speed up nearest neighbour subroutines. They all require an exponential amount of memory. Quantum speedups for enumeration have received significantly less attention. Hypothetical quadratic quantum speedups on enumeration were first suggested in [Sch+17]. Aono, Nguyen, and Shen [ANS18] demonstrated them by leveraging quantum backtracking techniques [Mon18; AK17] on the

[Reg05] Regev, "On lattices, learning with errors, random linear codes, and cryptography"

[Reg09] Regev, "On lattices, learning with errors, random linear codes, and cryptography"

[Sch+22] Schwabe et al., *CRYSTALS-KYBER*

[SE94] Schnorr and Euchner, "Lattice Basis Reduction: Improved Practical Algorithms and Solving Subset Sum Problems"

[LN20] Li and Nguyen, *A Complete Analysis of the BKZ Lattice Reduction Algorithm*

[Alb+21] Albrecht et al., "Lattice Reduction with Approximate Enumeration Oracles - Practical Algorithms and Concrete Performance"

[APS15b] Albrecht, Player, and Scott, "On the concrete hardness of Learning with Errors"

[Sch+17] Schanck et al., *NTRU-HRSS-KEM*

[ANS18] Aono, Nguyen, and Shen, "Quantum Lattice Enumeration and Tweaking Discrete Pruning"

[Mon18] Montanaro, "Quantum-Walk Speedup of Backtracking Algorithms"

[AK17] Ambainis and Kokainis, "Quantum algorithm for tree size estimation, with applications to backtracking and 2-player games"

enumeration tree constructed internally as part of enumeration. Bai, van Hoof, Johnson, Lange and Ngu [Bai+23] investigated concrete implementations of the arithmetic quantum circuits required.

[Bai+23] Bai et al., "Concrete Analysis of Quantum Lattice Enumeration"

While applicability of these speedups appears clear in the unbounded-depth logical-qubit model, where quantum computation achieves low error rates for free and does not decoherence, our current understanding of quantum computer engineering suggests that this model may be overly optimistic for hypothetical real-world quantum adversaries [Pre18]. For example, Albrecht, Gheorghiu, Postletwaite and Schanck [Alb+20b] investigate the impact of error correction on quantum lattice sieving, determining that achieving even small speedups over classical sieving in the cryptanalytic regime requires making several optimistic algorithmic and physical assumptions. We are currently not aware of any similar work on the validity of quantum speedups on enumeration in similarly constrained models.

[Pre18] Preskill, "Quantum Computing in the NISQ era and beyond"

[Alb+20b] Albrecht et al., "Estimating Quantum Speedups for Lattice Sieves"

Objective & Contribution. In this paper, we set to investigate whether quantum speedups on lattice enumeration [ANS18] apply to enumeration with extreme cylinder pruning in the *limited quantum depth* setting. In this setting, we assume the availability of error-corrected logical qubits and QRACM [GLM08; Kup11; JR23]. However, we also assume a limit MaxDepth to the depth that a quantum circuit can achieve, as computations with higher depth than MaxDepth are assumed to decohere, returning noise.

[ANS18] Aono, Nguyen, and Shen, "Quantum Lattice Enumeration and Tweaking Discrete Pruning"

[GLM08] Giovannetti, Lloyd, and Maccone, "Quantum Random Access Memory"

[Kup11] Kuperberg, *Another subexponential-time quantum algorithm for the dihedral hidden subgroup problem*

[JR23] Jaques and Rattew, *QRAM: A Survey and Critique*

[Nat17] National Institute for Standards and Technology, *Post-Quantum Cryptography Call for Proposals*

This setting has been proposed by the NIST in their call for proposals for post-quantum KEMs and digital signatures [Nat17, Sec. 4.A.5]. NIST proposes different values for MaxDepth (namely, of 2^{40}, 2^{64} and 2^{96}), capturing different run-times and quantum computing technology, and proposes costs for key-search attacks against block ciphers and collision-search attacks for hash functions in this setting. As a case-study, we adopt the same MaxDepth limitations, and investigate their effect on quantum enumeration with cylinder pruning against CRYSTALS-Kyber, the KEM selected for standardization by NIST.

Since our initial results suggest that enumeration trees constructed when attacking Kyber are mostly too large to be directly enumerated quantumly when MaxDepth is considered, we propose a combined classical-quantum enumeration algorithm that allows leveraging any available quantum computation capabilities, regardless of quantum depth budget limits. We provide a detailed yet generous-to-the-adversary analysis of the runtime costs of this combined attack in terms of quantum depth and number of gates under reasonable heuristics that we support with experimental evidence. We identify multiple known unknowns that affect the cost of the combined attack, and provide lower bounds for each one where possible, and otherwise provide experiment-backed heuristics. Finally, we use our analysis to estimate lower bounds on the cost of performing the primal lattice attack on Kyber in various settings, using our combined classical-quantum extreme cylinder pruning enumeration.

Our results suggest that quantum cylinder pruning enumeration techniques are unlikely to affect larger parameters sets for lattice-based schemes

when taking into consideration a MAXDEPTH constraint. While their effect on smaller parameters cannot be fully excluded, successful attacks are contingent on various unknown quantities being favorable to the attacker.

As part of our analysis, we also develop some minor results concerning the structure of lattice enumeration trees, which we report in the full version of this paper [Bin+23] together with matching experiments, and that are of independent interest. We have made the source code used to produce our experimental results, tables, and plots publicly available.

[Bin+23] Bindel et al., *Quantum Lattice Enumeration in Limited Depth*

OUTLINE. In Section 7.1 we describe our lower bounds on the cost of combined classical-quantum enumeration. In Section 7.3 we use our estimates to investigate the cost of quantum enumeration as part of the primal lattice attack on Kyber.

7.1 ESTIMATING THE COST OF QUANTUM ENUMERATION

In this section, we outline the components of our cost estimation of quantum lattice enumeration via backtracking under a MAXDEPTH restriction. We start by reviewing the gate-cost of Montanaro's FINDMV algorithm (cf. Section 3.1.2) and the depth of the quantum walk QPE($\mathcal{W}$), since the latter will imply an upper bound to the size of the largest tree that can be searched within a MAXDEPTH budget (cf. Section 3.4) for coherent quantum computations. We then proceed to explore the cost of combining quantum enumeration with classical one, to address settings where the trees are too large for the limited quantum depth budget.

7.1.1 *Quantum Backtracking to Find a Marked Vertex*

We follow the proof of Theorem 2 from Section 3.1.2, aiming to provide concrete lower bounds (rather than asymptotic upper bounds) to the cost of the FINDMV algorithm. The quantum backtracking framework laid out in Section 3.1.2 performs a classical depth-first search on the backtracking tree, where each node is evaluated using multiple, individual quantum walks to decide whether it spawns a subtree containing a marked node. Each quantum walk has *low* success probability, and the results of all walks are combined using a Chernoff bound to provide DETECTMV with a targeted success probability. An overview of Montanaro's quantum algorithm is sketched in Figure 7.1. Since the depth of a quantum circuit is the principal limitation for our cost model (see Section 3.3.2) all calls to the quantum circuit QPE($\mathcal{W}$) can be viewed independently, meaning their depth does not accumulate towards the MAXDEPTH limit.

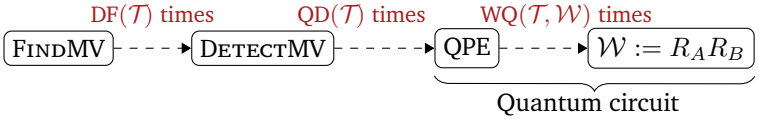


FIGURE 7.1: Overview of Montanaro's marked vertex (FINDMV) finding backtracking algorithm [Mon18]. Only the QPE part of the algorithm needs to run within MAXDEPTH.

Node Degree.

While Theorem 2 assumes the tree being enumerated having constant degree, this is not the case for enumeration trees (of depth n) on general lattices where the leaves are lattice vectors of norm at most R. Given a node $(\star, \dots, \star, c_{n-k+1}, \dots, c_n)$ on level k of the tree, an upper bound $\mathcal{C}_k$ on its degree corresponds to an upper bound on the number of possible values $c_{n-k} \in \mathbb{Z}$ such that $(\star, \dots, \star, c_{n-k}, c_{n-k+1}, \dots, c_n)$ is in the backtracking tree. An upper bound on the degree of the tree would then be $\mathcal{C} = \max_k \mathcal{C}_k$. For q-ary lattices as considered in our experiments in Section 7.3, a bound could be $\mathcal{C}_k = q$ for all k. A better bound is given by [Bin+23, Lemma 1], where we show $\mathcal{C}_k \approx \min(\lfloor 2 \cdot R_{k+1} / \|b^*_{n-k}\| \rceil, q)$.

[Bin+23] Bindel et al., *Quantum Lattice Enumeration in Limited Depth*

Any tree $\mathcal{T}$ of depth n and degree d can be transformed into a binary tree such that the resulting tree $\mathcal{T}'$ has depth $n' \leq n \log d$ and at most $\#\mathcal{T} \leq \#\mathcal{T}' \leq 2 \cdot \#\mathcal{T}$ nodes. Since we aim to provide a lower bound on quantum enumeration via backtracking and the quantum cost depends on the treesize, we do not consider a implicit transformation to a binary tree as input for Theorem 2. We do, however, consider a classical binary search in FindMV, since this is optimal in the classical case.

The result in [Mon18], however, also extends to trees with non-constant degree, as [Mon18, Sec. 1] notes that this assumption was made to simplify the complexity bounds. In particular, the assumption of a constant degree allows to bound the asymptotic cost of operator $\mathcal{W}$ and it simplifies the bound the asymptotic number of repetitions of DetectMV and thus of QPE($\mathcal{W}$).

[Mon18] Montanaro, "Quantum-Walk Speedup of Backtracking Algorithms"

Procedures.

As outlined in Figure 7.1, the quantum algorithm that can identify marked vertices in a tree, FindMV, will internally call DetectMV, which detects whether a marked vertex exists in a tree at all. This, in turn, runs quantum phase estimation QPE on the operator $\mathcal{W}$. We may refer to QPE($\mathcal{W}$) as *quantum walk*.

Each procedure calls the respective sub-procedure multiple times (with the number of calls depending on the properties of the respective tree $\mathcal{T}$), resulting in total depth and gate cost for FindMV of

$$\text{T-Depth}(\text{FindMV}(\mathcal{T})) = \text{WQ}(\mathcal{T}, \mathcal{W}) \cdot \text{T-Depth}(\mathcal{W}), \tag{7.1}$$

$$\text{GCost}(\text{FindMV}(\mathcal{T})) = \text{DF}(\mathcal{T}) \cdot \text{QD}(\mathcal{T}) \cdot \text{WQ}(\mathcal{T}, \mathcal{W}) \cdot \text{GCost}(\mathcal{W}), \tag{7.2}$$

where DF($\cdot$), QD($\cdot$), and WQ($\cdot$) are the number of calls to the subroutines DetectMV in FindMV, QPE in DetectMV, and $\mathcal{W}$ in QPE, respectively. We will analyze the number of these calls under the assumption that the search tree is of depth n and degree bound by $\mathcal{C}$, prioritizing strict lower bounds.

Number of Calls DF($\cdot$). Every call to FindMV (cf. Section 3.1.2) performs a classical search upon input tree $\mathcal{T}$, and outputs a single leaf on level n. Aono, Nguyen and Shen [ANS18] analyze the number of calls to DetectMV made when searching an enumeration tree without an asymptotically constant degree. Their analysis performs an asymptotically convenient implicit transformation of the tree into a binary tree [ANS18, Theorem 5], resulting in $O(n \log \mathcal{C})$ calls in the worst case, where we could take

[ANS18] Aono, Nguyen, and Shen, "Quantum Lattice Enumeration and Tweaking Discrete Pruning"

[Bin+23] Bindel et al., *Quantum Lattice Enumeration in Limited Depth*

$\mathcal{C} = \max_k\{\min(\lfloor 2 \cdot R_{k+1}/||b^*_{n-k}||\rceil, q)\}$ (cf. [Bin+23, App. C]). This bound is likely tight on average when trees are guaranteed to contain a marked leaf. However, as we will see in Section 7.1.3, a MaxDepth constraint results in performing quantum walks on (sub-)trees without such a guarantee. Since every lattice has a shortest vector, we assume that there exist at least one marked leaf in the enumeration tree. However, not every subtree of the enumeration tree will contain a marked leaf. For such subtrees, DetectMV is only called once with the output "no marked node exists". Since FindMV requires a single call to DetectMV to identify that a tree has no marked leafs, we lower bound $\mathrm{DF}(\cdot) \geq 1$. The success probability of a call to FindMV depends on that of DetectMV. In the following paragraph we lower bound the cost of DetectMV with success probability 1, so that FindMV also has full success probability.

Number of Calls $\mathrm{QD}(\cdot)$. An upper bound on the quantum depth required to run DetectMV can be established directly from [Mon18, Alg. 2] and [Mon18, Lemma 2.4].

[Mon18] Montanaro, "Quantum-Walk Speedup of Backtracking Algorithms"

Corollary 6 (T-Depth of quantum circuit to detect a marked node)**.** *Let $\mathcal{W}$ be a quantum operator of depth T-Depth$(\mathcal{W})$ that acts on a backtracking tree $\mathcal{T}$ in n (unassigned) variables. Let QPE$(\mathcal{W})$ be the quantum circuit performing phase estimation on $\mathcal{W}$. For any failure probability $\delta_{DMV} \in (0, 1)$, there exists a quantum algorithm DetectMV that decides with probability at least $1-\delta_{DMV}$ if a marked node exists in $\mathcal{T}$ by calling QPE $\lceil \epsilon \log(1/\delta_{DMV})\rceil$ times, such that T-Depth(QPE) $\leq 1/b\sqrt{\#\mathcal{T} \cdot n} \cdot$ T-Depth$(\mathcal{W})$, for some value $b > 0$ depending on $\mathcal{W}$.*

As mentioned in Section 3.1.2 and Corollary 6, each call to DetectMV repeats the QPE a total of $\lceil \epsilon \log(1/\delta_{\mathrm{DMV}})\rceil$ times for some constant $\epsilon > 0$. The value of ϵ depends on the failure probability of the quantum phase estimation QPE$(\mathcal{W})$, and on the desired failure probability δ_{DMV} of DetectMV. The failure probability of QPE$(\mathcal{W})$ in turn depends on the number of applications of the operator $\mathcal{W}$, relative to the tree-size $\#\mathcal{T}$, the dimension n of the lattice and a constant b (cf. Corollary 6). It is important to note that there is a trade-off between the number of repetitions of $\mathcal{W}$ in the QPE, and the number of repetitions of the QPE. We do not consider any optimizations related to this trade-off as they are implementation specific. Instead, since we are determining lower bounds for the number of calls, with $\epsilon \geq 0$ we lower bound $\mathrm{QD}(\mathcal{T}) \geq 1$.

Remark 10 (On the tightness of Corollary 6.)**.** *In the black-box setting, with query access to a predicate P, $\Omega(\sqrt{\#\mathcal{T}n})$ is a lower bound on the query-complexity to detect a marked node in a tree with $\#\mathcal{T}$ nodes and depth n, cf. [AA03, Theorem 7], [Mon18, Sec 4]. As such [Mon18] notes that [Mon18, Thm 1.1] and thus [Mon18, Lem 2.4] are optimal for $\delta = \Omega(1)$. As a consequence, Corollary 6 is a lower bound if $b \in \Omega(1)$, where in the black box setting T-Depth$(\mathcal{W}) = \Omega(1)$.*

[AA03] Aaronson and Ambainis, "Quantum Search of Spatial Regions"

Number of Calls $\mathrm{WQ}(\cdot)$. As used inside of DetectMV, QPE needs to be run with precision $b/\sqrt{\#\mathcal{T} \cdot n}$, for some constant $b > 0$, returning after $\approx \sqrt{\#\mathcal{T} \cdot n}/b$ evaluations of $\mathcal{W}$. Or put differently, asymptotically, $\Omega(\sqrt{\#\mathcal{T}n})$

is a lower bound on the query-complexity of detecting a marked node in a tree with $\#\mathcal{T}$ nodes and depth n [AA03, Theorem 7][Mon18, Sec 4]. Montanaro also notes that Thm. 1.1 and thus Lem. 2.4 of [Mon18] are optimal for $\delta_{\text{DMV}} \in \Omega(1)$. As a consequence, Corollary 6 is an asymptotic lower bound if $b \in \Omega(1)$, where in the black box setting T-DEPTH$(\mathcal{W}) \in \Omega(1)$.

[AA03] Aaronson and Ambainis, "Quantum Search of Spatial Regions"

[Mon18] Montanaro, "Quantum-Walk Speedup of Backtracking Algorithms"

Finding the hidden constant for the phase estimation is more involved. While explicit constants exist for phase estimation [Bes05], Montanaro's algorithm may not necessarily use the optimal majority voting scheme as part of DETECTMV. Notably, there is a trade-off between the number of repetitions of $\mathcal{W}$ in the QPE, and the number of repetitions of the QPE (cf. Section 7.1.2). Fewer repetitions of the first (reducing both T-DEPTH and GCOST) results in an increase of the number of repetitions of the QPE (increasing only GCOST). We do not consider optimizations related to this trade-off. While we investigate a possible approximation to the constants in Section 7.1.2, resulting in $b \leq 1/64$, for the sake of a lower bound we settle for the following conjecture for our estimations in Section 7.3, believing that this should not cause overestimating significantly the runtime of the attack given all the other strict lower bounds we adopt.

[Bes05] Bessen, "Lower bound for quantum phase estimation"

Conjecture 1. *The query complexity WQ$(\mathcal{T}, \mathcal{W})$ of quantum phase estimation of $\mathcal{W}$ (QPE$(\mathcal{W})$) is WQ$(\mathcal{T}, \mathcal{W}) \geq \sqrt{\#\mathcal{T} \cdot n}$.*

7.1.2 *Beyond Lower Bounds for DF, QD, WQ*

The lower bounds on DF$(\mathcal{T})$, QD$(\mathcal{T})$ and WQ$(\mathcal{T})$ result in a conservative cost estimation at the cost of potentially underestimating the attack cost. In this section, we discuss alternative lower bounds for the quantities representing the number of calls of the FINDMV and DETECTMV (cf. Figure 7.1).

We perform a heuristic analysis of the hidden constants, estimating tighter bounds for these quantities. Indeed, we show that the number of calls to DETECTMV is likely DF$(\mathcal{T}) \in \{1, n \log \mathcal{C}\}$ depending on the subtree being searched. Then a sufficient number of calls to the QPE in DETECTMV could be QD$(\mathcal{T}) = \lceil \epsilon \cdot \log(n \log(\mathcal{C})) \rceil$ with $\epsilon = 20$, and a sufficient number of calls to $\mathcal{W}$ during QPE is WQ$(\mathcal{T}) \approx 64\sqrt{\#\mathcal{T} n}$, adopting a constant $b \geq 1/64$.

While our estimations seem realistic, and support Conjecture 1, for the sake of keeping our analysis as conservative as possible, we will keep using the strict lower bounds above during attack cost estimation in Sec. 7.1.4 and 7.3. Analogue results to those in Section 7.3 using the less strict estimates can be found in Section 7.3.2.

A MORE CONSERVATIVE BOUND ON DF$(\mathcal{T})$. As mentioned in Section 7.1.1, the analysis in [ANS18] assumes an implicit transformation of $\mathcal{T}$ into a binary tree of depth $h \log \mathcal{C}$. DETECTMV is then called on the root level, in order to detect whether marked vertices are in the tree. A binary search as assumed by Aono, Nguyen and Shen [ANS18] can be implemented by recursively dividing the search space into smaller subsets, i. e., considering the subsets $\{1, 2, ..., \lfloor (\mathcal{C} - 1)/2 \rfloor\}$ and $\{\lceil \mathcal{C}/2 \rceil, ..., \mathcal{C}\}$, of which each "half" is divided again, to reduce the search space of the child nodes. Accordingly, the search space of the child nodes is reduced. The transformed tree $\mathcal{T}'$ with binary degree has $n' = n\lceil \log \mathcal{C} \rceil$ levels and at most $\#\mathcal{T}' \leq 2 \cdot \#\mathcal{T}$ nodes. Then

[ANS18] Aono, Nguyen, and Shen, "Quantum Lattice Enumeration and Tweaking Discrete Pruning"

finding a marked leaf requires to run through at least n' (i. e., if on each branching of the binary search the branch corresponding to the marked leaf is searched first) and at most $2n'$ nodes in the binary search (i. e., if on each branching of the binary search the branch corresponding to the marked node is searched second), where on each branching DETECTMV decides if the branch is pruned or not, i. e.,

$$n\lfloor \log \mathcal{C} \rfloor \leq \mathrm{DF}(\mathcal{T}) \leq 2n\lceil \log \mathcal{C} \rceil.$$

However, this would only be required for the subtree that contains the marked vertex. If no marked vertices are found, no more calls are required, and $\mathrm{DF}(\mathcal{T}) = 1$. If there are, further calls are made to identify the path from the root to the marked leaf, akin to the binary search. In total, in order to identify one marked leaf in $\mathcal{T}$ (or return error),

$$\mathrm{DF}(\mathcal{T}) = \begin{cases} 1, & \text{if } \mathcal{T} \text{ contains no marked leaves,} \\ h \log \mathcal{C}, & \text{if } \mathcal{T} \text{ contains at least one marked leaf.} \end{cases}$$

This discussion implicitly assumes the correctness of DETECTMV, which however can return incorrect results, increasing the complexity of an estimation on DF. One option would be running DETECTMV multiple times per level, with the amount of repetition depending on analysis of the specific DETECTMV implementation, as well as the noise model of the quantum computer.

A simpler analysis would be to estimate the failure probability of FINDMV when calling DETECTMV once per level, assuming a tight failure probability upper bound for DETECTMV($\mathcal{T}$) of δ_{DMV}. Then, the success probability of FINDMV would be about $(1-\delta_{\mathrm{DMV}})^{\mathrm{DF}(\mathcal{T})}$, which is $O(1)$ if $\delta_{\mathrm{DMV}} \approx 1/\mathrm{DF}(\mathcal{T})$[1]. Since *a priori* we do not know whether $\mathcal{T}$ contains a marked vertex, this would mean implementing DETECTMV such that $\delta_{\mathrm{DMV}} \approx (h \log \mathcal{C})^{-1}$, to account for the in-principle $h \log \mathcal{C}$ successful calls required to detect the marked vertex in its subtree. We proceed to do this next.

[1] Specifically, it approaches e^{-1} as $\delta_{\mathrm{DMV}} \to 0$.

A MORE CONSERVATIVE BOUND ON QD($\mathcal{T}$). The way DETECTMV [Mon18, Alg. 2] is computed is by performing multiple times, say K, QPE on the $\mathcal{W}$ operator.

[Mon18] Montanaro, "Quantum-Walk Speedup of Backtracking Algorithms"

Let X_i be a random variable valued 1 when the i^{th} call to QPE($\mathcal{W}$) returned an eigenvalue 1, and valued 0 otherwise, and let $Y_K = \sum_{i\in[K]} X_i$. The X_i are then independent and identically distributed Bernoulli random variables. Let MV denote the event that a marked vertex is contained in $\mathcal{T}$ and $\overline{MV}$ the opposite event. The core idea around DETECTMV is that whenever a marked vertex exists, $QPE(\mathcal{W})$ will tend to return 1, and 0 otherwise. Indeed, from [Mon18, Proof of Lemma 2.4] we have that $p_1 := \Pr[X_i = 1 \mid \overline{MV}] \leq 1/4$ and $p_2 := \Pr[X_i = 0 \mid MV] \leq 1/2$. In order to decide whether a tree contains a marked vertex, we run K instances of QPE on $\mathcal{W}$, and then check whether enough instances returned 1. Namely, for some fixed $\alpha \in (0, 1]$ to be determined, we return "marked vertex exists" if and only if $Y_K \geq \alpha K$.

As seen in the previous paragraph on DF($\mathcal{T}$), we may want to fix a target failure probability δ_{DMV} for DETECTMV, which can be achieved by

picking K high enough. We start assuming we have found α, and use Chernoff bounds to estimate upper bounds on the "false positive/negative" probabilities $\Pr[Y \geq \alpha K \mid \overline{MV}]$ and $\Pr[Y \leq \alpha K \mid MV]$.

By direct computation of the bound, recalling that the X_i are independent and identically distributed, we have

$$\begin{aligned}\Pr[Y_K \geq \alpha K \mid \overline{MV}] &\leq \exp(-t\alpha K)\mathbb{E}\left[\exp(tY_K)\right] \\ &= \exp(-t\alpha K)\mathbb{E}\left[\prod_{i=1}^{K}\exp(tX_i)\right] \\ &= \exp(-t\alpha K)\mathbb{E}\left[(\exp(tX_1))\right]^K \\ &= \left(\frac{1+p_1(\exp(t)-1)}{\exp(\alpha t)}\right)^K,\end{aligned}$$

for any $t > 0$. For $\Pr[Y_K \leq \alpha K]$ a similar computation on the left tail gives

$$\Pr[Y_K \leq \alpha K \mid MV] \leq \left(\frac{\exp(t)+p_2(1-\exp(t))}{\exp(\alpha t)}\right)^K, \text{ for any } t < 0.$$

We then identify values of α and ε such that

$$\inf_{y>0}\left(\frac{1+p_1(\exp(t)-1)}{\exp(\alpha t)}\right)^{\varepsilon} \leq 1/2 \tag{7.3}$$

and

$$\inf_{y<0}\left(\frac{\exp(t)+p_2(1-\exp(t))}{\exp(\alpha t)}\right)^{\varepsilon} \leq 1/2\,. \tag{7.4}$$

From a numerical search, and picking the smallest possible ε returned, we observe a valid pair at $\alpha = 0.369017$ and $\varepsilon = 20$.[2] Finally, we compute

$$\Pr[Y_K \geq \alpha K \mid \overline{MV}] \leq \inf_{y>0}\left(\frac{1+p_1(\exp(t)-1)}{\exp(\alpha t)}\right)^K \leq (1/2)^{K/\varepsilon},$$

and similarly for $\Pr[Y_K \leq \alpha K \mid MV]$, suggesting that to get an overall failure probability of at most δ_{DMV}, one should choose $K/\varepsilon \geq \log(1/\delta_{\text{DMV}})$. Therefore, it should be sufficient to choose

$$\text{QD}(\mathcal{T}) = K = \lceil 20\log(h\log\mathcal{C})\rceil \geq \varepsilon\log(1/\delta_{\text{DMV}}).$$

[2] In particular, our value of α is quite close to the $3/8 = 0.375$ proposed in [Mon18]. This latter value would however imply $\varepsilon = 22$.

A MORE CONSERVATIVE BOUND ON $\text{WQ}(\cdot)$. Let $\overline{MV}$ denote the event that no marked vertex is contained in the tree $\mathcal{T}$. An approximation for the constants in QPE was analyzed by [CKM19, Sec 4.1], who showed that the probability $p_1 \coloneqq \Pr[X_i = 1 \mid \overline{MV}]$ that the QPE outputs *one* even if no marked vertex exists is bounded by $p_1 \leq 2\sqrt{b}(1+o(1))$. In our case, setting p_1 as low as possible leads to $p_1 \leq 1/4$, which implies using at least $b \geq 1/64$. Since we evaluate $\mathcal{W}$ about $\sqrt{\#\mathcal{T}\cdot n}/b$ times and $1/b \leq 64$, it follows

$$\text{WQ}(\mathcal{T},\mathcal{W}) \approx 64\sqrt{\#\mathcal{T}\cdot n}\,.$$

Depth of QPE($\mathcal{W}$).

We turn back to the lower bounds estimated in Section 7.1.1. With Conjecture 1 in hand, we can attempt to estimate the depth budget required to break practical lattice-based schemes using quantum enumeration as proposed by

[CKM19] Campbell, Khurana, and Montanaro, "Applying quantum algorithms to constraint satisfaction problems"

[ANS18] Aono, Nguyen, and Shen, "Quantum Lattice Enumeration and Tweaking Discrete Pruning"

[APS15b] Albrecht, Player, and Scott, "On the concrete hardness of Learning with Errors"

[SE94] Schnorr and Euchner, "Lattice Basis Reduction: Improved Practical Algorithms and Solving Subset Sum Problems"

[Sch+22] Schwabe et al., *CRYSTALS-KYBER*

Aono, Nguyen and Shen [ANS18]. By using the `lwe-estimator` [APS15b] we obtain the block size β required by the BKZ [SE94; SE94] algorithm to successfully run key recovery on Kyber [Sch+22] using the primal lattice attack. Using $n = \beta$ and $\mathbb{E}[\#\mathcal{T}]$ equal to the returned cost of enumeration when using a custom cost model implementing the lower bound from [Aon+18, Eq. 16], and momentarily assuming $\mathbb{E}[\sqrt{\#\mathcal{T}}] \approx \sqrt{\mathbb{E}[\#\mathcal{T}]}$ (cf. Section 7.1.4), we can see that

[Aon+18] Aono et al., "Lower Bounds on Lattice Enumeration with Extreme Pruning"

$$\log \mathbb{E}[\sqrt{\#\mathcal{T}n}] \approx \frac{\log \mathbb{E}[\#\mathcal{T}] + \log \beta}{2} \approx \begin{cases} 90.3 & \text{for Kyber-512,} \\ 166.2 & \text{for Kyber-768,} \\ 263.7 & \text{for Kyber-1024}\,, \end{cases}$$

with $\mathcal{T}$ collecting 2^{64} cylinder pruning enumeration trees of dimension β.

While the above numbers are a rule-of-thumb approximation, it can be seen that most likely and regardless of the value of T-Depth($\mathcal{W}$), breaking Kyber-768 and Kyber-1024 with a single direct quantum enumeration will not be possible within MaxDepth $\leq 2^{96}$. To deal with this issue, we propose combining quantum backtracking with classical enumeration, in a manner similar to classical parallel enumeration.

7.1.3 *Combined Classical-Quantum Enumeration*

Generally, parallelization of lattice enumeration [DS10; Her+10; Kuo+11] is conceptually simple, as the tree structure directly induces a partitioning to the search problem. This means that when searching for short vectors on a tree n levels deep (where $n = \beta$ is the BKZ block size), one can first serially enumerate all nodes on level $k < n$, and then run in parallel lattice enumeration on the subtrees rooted at level k of depth at most $n - k$.

[DS10] Dagdelen and Schneider, *Parallel Enumeration of Shortest Lattice Vectors*

[Her+10] Hermans et al., "Parallel Shortest Lattice Vector Enumeration on Graphics Cards"

[Kuo+11] Kuo et al., "Extreme Enumeration on GPU and in Clouds - - How Many Dollars You Need to Break SVP Challenges -"

Following this approach, we will run classical enumeration up to depth k, and proceed to run quantum enumeration on the smaller subtrees of depth $h \leq r - k$, corresponding to sub-lattices of dimension h. We will choose k such that the depth of any call to QPE($\mathcal{W}$) is within the limit of MaxDepth, following Corollary 6. We depict the general strategy in Figure 7.2.

This combined approach is independent of the implementation of the quantum circuits, particularly of the operator $\mathcal{W}$. This means we would be able to estimate bounds on the cost of the attack given different possible values for T-Depth($\mathcal{W}$) and GCost($\mathcal{W}$), including generous lower bounds (cf. Section 7.2.1).

The approach is also compatible with pruned enumeration techniques. In the remainder of the paper, we will focus in particular on cylinder pruning [GNR10]. We will start by analyzing the cost of the combined enumeration algorithm on a single (possibly pruned) tree, in Section 7.1.4, and extend this to the case where M pruned trees are combined to achieve high success probability, in Section 7.1.5.

[GNR10] Gama, Nguyen, and Regev, "Lattice Enumeration Using Extreme Pruning"

7.1.4 *Combined Enumeration of a Single Tree*

We start by recalling some notation introduced in Section 4.3 to describe enumeration trees, illustrated in Figure 7.2. Given a tree of depth n, its nodes are partitioned into sets $(Z_i)_{i=1}^n$ of expected cardinality $(H_i)_{i=1}^n$ over

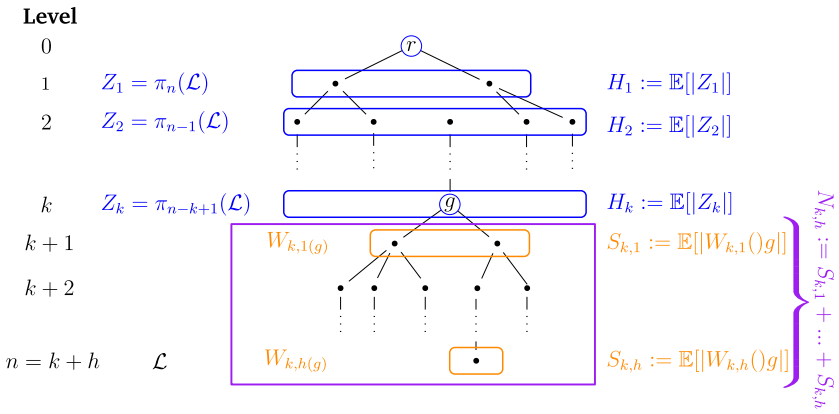


FIGURE 7.2: Combined classical-quantum enumeration tree. Quantum enumeration will happen on subtrees rooted at level k, here circled in purple.

the distribution of lattices being inspected, where Z_i collects all the nodes "on level i", that is, of distance i from the root node r. Any node $g \in Z_k$ generates a subtree $\mathcal{T}(g)$ of depth $h \leq n-k$. The nodes of this subtree are partitioned into sets $(W_{k,i}(g))_{i=1}^{h}$ of expected cardinality $(S_{k,i})_{i=1}^{h}$ over random trees and g distributed uniformly in Z_k. The expected number of nodes in the subtree $\mathcal{T}(g)$, including the root g, is $1 + N_{k,h}$, where $N_{k,h} := \sum_{i=1}^{h} S_{k,i}$, while the expected number of nodes in the entire enumeration tree $\mathcal{T}$, including the root r, equals $1 + \frac{1}{2}\sum_{i=1}^{n} H_i$. The $\frac{1}{2}$ factor comes from the fact that the tree is symmetric around 0, and hence only half of the tree needs to be searched to identify all the vectors within the enumeration radius, up to sign.

Classically and Quantumly Enumerated Components.

We first discuss how we divide the classical and quantum components of the enumeration algorithm.

CLASSICAL COMPONENT. In the setting of combined classical-quantum enumeration, let k be the level up to where classical enumeration is performed. This costs

$$\mathop{\mathbb{E}}_{\substack{\text{random}\\ \text{tree } \mathcal{T}}}[\text{Classical GCOST}] \approx 1 + \frac{1}{2}\sum_{i=1}^{k} H_i, \tag{7.5}$$

where we equate the cost of classical enumeration to the number of nodes visited by the algorithm, as it is standard in the literature.

QUANTUM COMPONENT. After the classical enumeration phase, we have H_k nodes on level k, each admitting a subtree of height $h \leq n-k$, and covering the remaining levels of the full enumeration tree. A natural approach could be to enumerate all H_k subtrees individually (and possibly in parallel). However, it is not known which subtree contains the (likely few) marked nodes, meaning that we would be running $\approx H_k$ calls to quantum enumeration. In both pruned and non-pruned enumeration, the bulk of the nodes contained in the trees being traversed is contained in the "middle" levels Z_i for $i \approx n/2$, with pruning "spreading the bulk" on a larger window of levels around $n/2$ [GNR10, Fig. 1] as in Figure 7.3. For our setting, this

[GNR10] Gama, Nguyen, and Regev, "Lattice Enumeration Using Extreme Pruning"

would imply three scenarios, depending on k:

$k \approx 1$, in this case most of the tree fits within the quantum enumeration subroutine, and a quadratic speedup is possibly achievable,

$k \approx n/2$, in which case we would be running $\approx H_{n/2}$ quantum enumeration calls, meaning that the GCOST of the operation would be proportional to $H_{n/2}$, which is approximately the cost of fully-classical enumeration.

$k \approx n$, which means that we would be running some quantum enumeration, but the bulk of the enumeration would anyway be classical, nullifying any possible speedups from quantum enumeration.

While the case $k \approx 1$ is possible, requiring to tolerate a high enough MAXDEPTH to traverse most of the enumeration tree within the quantum subroutine is quite restricting; in particular, in light of the rule-of-thumb numbers from Section 7.1.2.

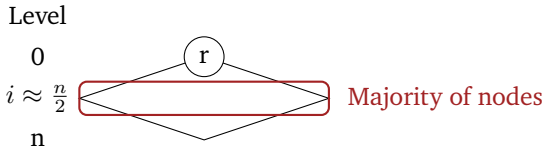


FIGURE 7.3: The bulk of the nodes contained in the enumeration trees being traversed is contained in the "middle" levels Z_i for $i \approx n/2$ [GNR10, Fig. 1].

A possible alternative approach for the quantum phase of the attack is to collect all the subtrees rooted on level k under a single tree, by adding a "virtual" root node as the "parent" to all the nodes in Z_k, and to run quantum enumeration on this tree; we illustrate this in Figure 7.4 for multiple virtual nodes. This approach has the advantage of running a single quantum enumeration rather than H_k, potentially achieving a better speedup than in the previous case. However, this comes at a cost in terms of QRACM [GLM08; Kup11; JR23], since the $g_i \in Z_k$ would need to be first enumerated and then stored in memory, to be accessible for the quantum algorithm. Except for $k \approx 1$ or $k \approx n$, this approach may require a super-exponential amount H_k of QRACM for any meaningfully small value of k such that a speedup can be achieved (say, for $k \lessapprox n/2$).

Given the issues of the two methods above, we consider an interpolation of both. We assume to have access to enough QRACM to store at a time 2^y nodes on level k. We combine classical and quantum enumeration by using a classical enumeration routine to visit up to 2^y nodes $\{g_i\}_i \in Z_k$, and collect them under a *virtual* root node v as to form a tree $\mathcal{T}(v)$. We then run quantum enumeration on $\mathcal{T}(v)$. If we find a short leaf in $\mathcal{T}(v)$, we terminate. Otherwise, we resume classical enumeration and repeat the collection and the quantum enumeration processes.

Let $(g_i)_i = g_1, g_2, \cdots \in \mathcal{T}$ be the sequence of nodes in the full enumeration tree $\mathcal{T}$ visited in order by the classical enumeration. Let $g_{k_1}, g_{k_2}, \cdots \in Z_k$ be the subsequence of $(g_i)_i$ of nodes *on level* k. Let $S_1 = \{g_{k_1}, \ldots, g_{k_{2^y}}\}, S_2 = \{g_{k_{2^y+1}}, \ldots, g_{k_{2^y+2^y}}\}, \cdots \subset Z_k$ be the subsets of size 2^y that our combined classical-quantum enumeration routine collects under virtual nodes v_i, such that S_i is the set of nodes on the first

[GLM08] Giovannetti, Lloyd, and Maccone, "Quantum Random Access Memory"

[Kup11] Kuperberg, *Another subexponential-time quantum algorithm for the dihedral hidden subgroup problem*

[JR23] Jaques and Rattew, *QRAM: A Survey and Critique*

level of the subtree $\mathcal{T}(v_i)$. To be able to estimate the cost we make the following conjecture.

Conjecture 2. *Consider $V_i := \text{GCost}(\text{FindMV}(\mathcal{T}(v_i)))$ as random variables under the randomness of the distribution of enumeration trees for random lattices. Then the V_i are identically distributed.*

In [Bin+23, App. E] we present experimental evidence supporting this conjecture in the case of pruned enumeration. Under Conjecture 2, we estimate that the expected quantum gate cost of combined classical-quantum enumeration is approximately

[Bin+23] Bindel et al., *Quantum Lattice Enumeration in Limited Depth*

$$\begin{aligned}\mathop{\mathbb{E}}_{\substack{\text{random}\\ \text{tree } \mathcal{T}}}[\text{Quantum GCost}] &\approx \mathop{\mathbb{E}}_{\substack{\text{random}\\ \text{tree } \mathcal{T}}}\left[\sum_{\substack{v,\text{ out of the}\\ (1/2)\cdot H_k/2^y}} \text{GCost}(\text{FindMV}(\mathcal{T}(v)))\right] \\ &= \frac{1}{2}\cdot\frac{H_k}{2^y}\cdot\mathbb{E}[\text{GCost}(\text{FindMV}(\mathcal{T}(v)))] \quad \text{(by Wald's identity)}\,, \quad (7.6)\end{aligned}$$

where the factor of $1/2$ is due to the lattice's additive symmetry.

Expected Cost of one Quantum Enumeration.

After having illustrated how to divide the classical and quantum components, we move our attention to computing the expected gate-cost of FindMV on subtrees $\mathcal{T}(g)$ rooted at some node g. Here, we repurpose the analysis from Section 7.1.1, adapting it to the case where the enumerated tree is rooted on level k and has depth $h \le n-k+1$.[3]

[3] The "+1" term coming from adding a "virtual" root node.

We start by recalling Equation (7.2) for the gate-cost of FindMV for a tree $\mathcal{T}(g)$ of height h and with at most $\mathcal{C}$ children per node, following the analysis in Section 7.1.1, which is

$$\begin{aligned}\text{GCost}(\text{FindMV}(\mathcal{T}(g))) &= \text{DF}(\mathcal{T}(g))\cdot\text{QD}(\mathcal{T}(g))\cdot\text{WQ}(\mathcal{T}(g),\mathcal{W})\cdot\text{GCost}(\mathcal{W}) \\ &\ge \sqrt{\#\mathcal{T}(g)\cdot h}\cdot\text{GCost}(\mathcal{W}).\end{aligned}$$

The GCost of operator $\mathcal{W}$ and the expected number of repetitions are independent of each other, and thus, the respective cost can be analyzed individually. The cost of the former is explored in Section 7.2, while we elaborate on the number of repetitions for a single tree next.

To compute $\mathbb{E}\left[\text{GCost}(\text{FindMV}(\mathcal{T}(g)))\right]$ we first notice that $\text{DF}(\mathcal{T}(g))$ and $\text{QD}(\mathcal{T}(g))$ are constant quantities in our analysis (namely, we set both to 1), although in general they depend on the lattice problem and our setup of the full algorithm (such as in the choice of k, which would be done a priori based on cost estimations), as we describe in Section 7.1.2. Similarly, we set $h = n-k+1$. Hence, $\text{DF}(\mathcal{T}(g))$, $\text{QD}(\mathcal{T}(g))$, and h do not have a probability distribution, and do not affect the computation of the expectation. Similarly, the design of $\mathcal{W}$ is done *a priori*[4], and thus, the resulting $\text{GCost}(\mathcal{W})$ is a constant (cf. Section 7.2).

[4] Before applying the quantum operator $\mathcal{W}$, one has to define the circuit using an upper bound on the depth of the subtrees, since this depth cannot be determined from the root node alone. Since one of them will contain the marked vertex (and hence be of full height), we have to prepare the $\mathcal{W}$ circuit to tolerate traversing a full-height subtree. This is then constant for all calls to $\mathcal{W}$.

Remark 11. *We must note that if the enumeration bound R is small enough to guarantee only a few marked leaves in the full enumeration tree as in our case, then the subtrees $\mathcal{T}(g)$ will likely often contain no marked leaf, and hence be of*

height strictly smaller than $n-k+1$. On the one hand, this means that on most of the $\mathcal{T}(g)$ trees, DetectMV will be called only once at the root, meaning DF$(\mathcal{T}(g))=1$ (cf. Section 7.1.1). On the other hand, as part of WQ$(\mathcal{T}(g),\mathcal{W})$ we must nonetheless assume "full height" $h=n-k+1$, since the few trees containing marked leaves are of this height. Underestimating the tree height during QPE would mean that the marked leaves would not be found by FindMV. As we are aiming for strict lower bounds, we do not consider the full height of the implicit binary tree from Aono, Nguyen and Shen [ANS18].

[ANS18] Aono, Nguyen, and Shen, "Quantum Lattice Enumeration and Tweaking Discrete Pruning"

This leaves us with having to estimate $\mathbb{E}[\sqrt{\#\mathcal{T}(g)}]$. As pointed out in [ANS18], the probability distribution of the number of nodes in enumeration trees (or subtrees, such as $\mathcal{T}(g)$) is not known. Jensen's inequality gives an upper bound $\sqrt{\mathbb{E}[\#\mathcal{T}(g)]}$ but no clear lower bound. We address this by defining the *multiplicative Jensen's gap*, and evaluate the cost of the attack for different values of it.

Definition 7.1.1 (Multiplicative Jensen's gap)**.** *Let X be a random variable. We say X has multiplicative Jensen's gap 2^z if $\sqrt{\mathbb{E}[X]}=2^z\,\mathbb{E}[\sqrt{X}]$.*

This leaves us to having to estimate $\mathbb{E}[\#\mathcal{T}(g)]$. Given a "virtual" node g collecting 2^y subtrees rooted at nodes $\{g_1,\dots,g_{2^y}\}\subset Z_k$, by linearity of expectations

$$\begin{aligned}\mathop{\mathbb{E}}_{\mathcal{T},\{g_i\}_i}[\#\mathcal{T}(g)] &= 1+\sum_{i=1}^{2^y}\mathop{\mathbb{E}}_{\mathcal{T},\{g_i\}_i}[\#\mathcal{T}(g_i)]\\ &= 1+2^y\,(1+N_{k,h})\\ &= 1+2^y+2^y\sum_{j=1}^{h}S_{k,j},\end{aligned}$$

where the expectation is taken over the distribution of random trees $\mathcal{T}$, and $\{g_1,\dots,g_{2^y}\}$ is assumed to be as a set of random elements of Z_k. While this may not be exactly true, as it may be easier to find "related" elements in Z_k, where their coefficient vectors are similar, we believe this gives a good approximation of the cost.

To lower bound $S_{k,j}$, we start by observing that the $W_{k,j}(g)$ partition the set Z_{k+j}, since every element in the latter descends from an unique element $g\in Z_k$. By the definition of expectation,

$$S_{k,j}=\mathop{\mathbb{E}}_{\substack{g\sim U(Z_k)\\ \text{rand. }\mathcal{T}}}[|W_{k,j}(g)|]=\mathop{\mathbb{E}}_{\substack{\text{random}\\ \text{tree }\mathcal{T}}}\left[\sum_{g\in Z_k}\frac{|W_{k,j}(g)|}{|Z_k|}\right]=\mathop{\mathbb{E}}_{\substack{\text{random}\\ \text{tree }\mathcal{T}}}\left[\frac{|Z_{k+j}|}{|Z_k|}\right].$$

We then make a further conjecture to bound the expectation.

Conjecture 3. *Given a random enumeration tree generated as part of BKZ-β reduction for β large enough such that the Gaussian heuristic applies,[5] a level $k\geq 1$, and a node $g\in Z_k$, the expected number of nodes in level $k+j$ descending from g is $\mathbb{E}\left[\frac{|Z_{k+j}|}{|Z_k|}\right]\geq\frac{1}{2}\cdot\frac{\mathbb{E}[|Z_{k+j}|]}{\mathbb{E}[|Z_k|]}=\frac{1}{2}\cdot\frac{H_{k+j}}{H_k}$, where the expectation is taken over the distribution of random trees $\mathcal{T}$, and g is uniformly distributed in Z_k.*

[5] Experimentally, $\beta>45$ appears to be sufficient.

In [Bin+23, App. B] we provide experimental evidence supporting Conjecture 3. In [Bin+23, App. C] we further observe that in practice often the stronger approximation $S_{k,j}\approx H_{k+j}/H_k$ holds.

[Bin+23] Bindel et al., *Quantum Lattice Enumeration in Limited Depth*

Combining all the steps above gives us a heuristic estimation of

$$\begin{aligned}&\mathop{\mathbb{E}}_{\substack{\text{random}\\ \text{tree } \mathcal{T}}}[\text{Quantum GCost}] \approx \frac{1}{2}\cdot\frac{H_k}{2^y}\cdot\mathbb{E}\left[\text{GCost}(\text{FindMV}(\mathcal{T}(g)))\right]\\ &\geq \frac{1}{2}\cdot\frac{H_k}{2^y}\left(\frac{1}{2^z}\sqrt{\left(1+2^y+2^{y-1}\sum_{j=1}^{n-k+1}\frac{H_{k+j}}{H_k}\right)(n-k+1)}\right)\text{GCost}(\mathcal{W}),\end{aligned} \tag{7.7}$$

reducing the estimation of a lower bound on the expected cost to the estimation of H_i and $\text{GCost}(\mathcal{W})$. The estimation of H_i can be done using standard lattice cryptanalysis techniques; we provide a detailed derivation in the case of no pruning and of extreme cylinder pruning in [Bin+23, App. A]. Estimations for $\text{GCost}(\mathcal{W})$ are discussed in Section 7.2.1.

[Bin+23] Bindel et al., *Quantum Lattice Enumeration in Limited Depth*

The same approach can be used to determine the depth of quantum phase estimation of $\mathcal{W}$, which is the limiting factor in a runtime analysis with limited depth budget, resulting in

$$\begin{aligned}&\text{T-Depth}(\text{QPE}(\mathcal{W}))\\ &\geq \frac{1}{2^z}\sqrt{\left(1+2^y+2^{y-1}\sum_{j=1}^{n-k+1}\frac{H_{k+j}}{H_k}\right)(n-k+1)}\cdot\text{T-Depth}(\mathcal{W}).\end{aligned} \tag{7.8}$$

Remark 12 (On a more conservative bound on $\text{DF}(\mathcal{T})$)**.** *Now we can review a better bound on $DF(\mathcal{T})$ as in Section 7.1.2. In the setting of combined classical-quantum enumeration, most of the $H_k/2^y$ trees $\mathcal{T}(g)$ explored will not contain any marked leafs. Given that the quantum GCost is estimated (cf. Equation (7.7)) as*

$$\begin{aligned}&\mathop{\mathbb{E}}_{\substack{\text{random}\\ \text{tree } \mathcal{T}}}[\textit{Quantum GCost}] \approx \frac{H_k}{2^{y+1}}\cdot\mathbb{E}\left[\textit{GCost}(\textit{FindMV}(\mathcal{T}(g)))\right]\\ &= \frac{H_k}{2^{y+1}}\cdot\mathbb{E}[\textit{DF}(\mathcal{T}(g))\cdot\textit{QD}(\mathcal{T}(g))\cdot\textit{WQ}(\mathcal{T}(g),\mathcal{W})\cdot\textit{GCost}(\mathcal{W})],\end{aligned}$$

we could set $DF(\mathcal{T}(g)) = \left(\frac{H_k}{2^{y+1}} - 1 + h\log\mathcal{C}\right)\cdot\frac{2^{y+1}}{H_k}$ with $h = n-k+1$ during cost estimation, to capture how when the enumeration radius is short enough, $h\log\mathcal{C}$ calls will likely be made on only one of the subtrees, and one call will be made to the remaining $\frac{H_k}{2^{y+1}} - 1$ subtrees.

7.1.5 *Combined Enumeration of a Set of Trees*

In the context of enumeration with extreme pruning [GNR10] one considers a trade-off between the success probability of finding a short vector and the number of nodes pruned. Let p be the probability that the enumeration tree contains a node corresponding to a sufficiently short lattice point, that is $p = 1$ if the full tree is enumerated and $p < 1$ if branches are pruned. In the case of extreme pruning, $p \ll 1$, meaning that enumeration of the tree is much cheaper, but likely to fail. To boost the success probability, the original lattice basis is re-randomized M times, for some large value of M. Under assumptions of independence between the resulting re-randomized pruned trees, the probability of finding a short vector in at least one of the

[GNR10] Gama, Nguyen, and Regev, "Lattice Enumeration Using Extreme Pruning"

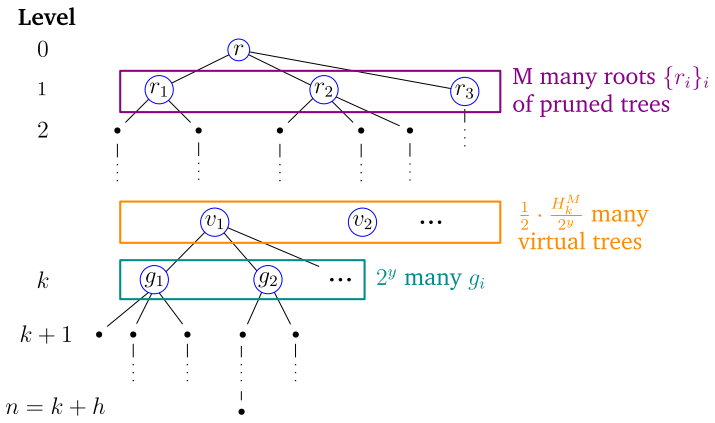


FIGURE 7.4: Full enumeration tree $\mathcal{T}^M$ for combined classical-quantum enumeration. Nodes and links in blue correspond to "virtual" nodes.

M pruned trees is $1-(1-p)^M = 1-(1-Mp+O(p^2))$, which is high if $M \approx 1/p$ as $M \gg 1$. It should be noticed that in practice re-randomization usually lowers the quality of the bases, essentially "undoing" some of the lattice reduction [Alb+20a, Sec. 2.5]. However, we will ignore this effect for the sake of finding a simple lower bound and assume that the quality of re-randomized bases for the same lattice is the same as the one for the original basis. Moreover, we assume that pruned trees corresponding to these re-randomized bases are independent of each other.

For our purposes, we will collect the M trees corresponding to M bases into a single tree $\mathcal{T}^M$, by adding a top "super-tree" connecting their roots to an overall root. Let r be the root node of this new tree and let $r_1, r_2, ..., r_M$ be the root nodes of the enumeration trees with the re-randomized bases $(B_i)_{i=1}^M$. We arrange r as parent node of the r_i; a sketch of the full tree is illustrated in Figure 7.4. The backtracking predicate (cf. Section 3.1.2) that decides on branching on input of a node r_i will always return INDETERMINATE on the levels 0 and 1, since all enumeration subtrees rooted at the respective r_i are independent of each other due to basis re-randomisation. We define a quantity H_k^M counting the expected number of nodes on level k of $\mathcal{T}^M$, in terms of $H_k^{(i)} := \mathbb{E}[\#\{\text{nodes on level } k \text{ of } \mathcal{T}(r_i)\}]$ (that is "H_k from tree $\mathcal{T}(r_i)$"), where $\mathcal{T}(r_i)$ is the pruned enumeration tree rooted at r_i. It follows that

$$\begin{aligned} H_0^M &= 1\,, \\ H_1^M &= M\,, \\ H_k^M &= \sum_{i\in[M]} H_{k-1}^{(i)} = M \cdot H_{k-1} \quad \textbf{if } k > 1\,. \end{aligned}$$

From H_k^M we can then redefine $S_{k,j}^M$ similarly to $S_{k,j}$, reducing it to H_{k-1}. This means that we can "port" our cost formulae from Section 7.1.4 by replacing H_k with H_k^M. We estimate H_k^M in the cases of no pruning and of extreme cylinder pruning in [Bin+23, App. A] since this is a standard computation taken from [GNR10; Aon+18], and continue with the cost estimation GCOST$(\mathcal{W})$ in the next subsection.

[Alb+20a] Albrecht et al., "Faster Enumeration-Based Lattice Reduction: Root Hermite Factor $k^{1/(2k)}$ Time $k^{k/8+o(k)}$"

[Bin+23] Bindel et al., *Quantum Lattice Enumeration in Limited Depth*

[GNR10] Gama, Nguyen, and Regev, "Lattice Enumeration Using Extreme Pruning"

[Aon+18] Aono et al., "Lower Bounds on Lattice Enumeration with Extreme Pruning"

7.2 INSTANTIATIONS FOR THE QUANTUM OPERATOR $\mathcal{W}$

In this section, we focus on exploring lower bound costs for the quantum operator $\mathcal{W}$. At its core, quantum enumeration consists of multiple repetitions of QPE on the operator $\mathcal{W}$ (see Figure 7.1), meaning that concrete estimates of the depth and gate-cost of quantum enumeration depend on the size of $\mathcal{W}$. In estimating whether quantum enumeration could be leveraged under a MAXDEPTH constraint, $\mathcal{W}$ plays two roles. First, its depth T-DEPTH($\mathcal{W}$) plays a part in determining how much classical *versus* quantum enumeration is used, by constraining the admissible values for k, y and z (cf. Equation (7.8)) based on the requirement that T-DEPTH(QPE($\mathcal{W}$)) $\leq$ MAXDEPTH, since by Corollary 6 the gate depth of QPE is partly determined by T-DEPTH($\mathcal{W}$). Second, its gate-cost GCOST($\mathcal{W}$) is a factor in estimating the total cost of the attack.

7.2.1 *Query-based Model*

The query-based model (cf. Section 3.3.1) assumes access to a black-box oracle that computes on-demand the operator $\mathcal{W}$ on any input. Therefore, we account any query to the operator as having "unit cost", meaning T-DEPTH($\mathcal{W}$) = GCOST($\mathcal{W}$) = 1.This setting implies a conservative lower bound on the cost of quantum enumeration. It also represents the setting where classical-quantum enumeration can make the most of any hypothetical quantum speedups. As such, the resulting gate depth is a very conservative *lower* bound as implementations of the operator $\mathcal{W}$ are of depth greater than one.

7.2.2 *Circuit-based Model*

Here, we aim to estimate the size of a *minimal* or *smallest known* quantum operator $\mathcal{W}$ in the circuit-based model (cf. Section 3.3.2). The objective is to provide a more realistic lowest-known bound on the gate cost (i. e., number of T gates) of $\mathcal{W}$ than the very conservative query-based model. Our focus is on finding an approximate smallest T-count and T-depth for arithmetic operations used inside of $\mathcal{W}$. We claim that, at a minimum, the operator $\mathcal{W}$ has to implement a predicate P deciding whether a projected lattice point has norm larger than the pruning bound R_i. We assume that each coefficient of the state vector $|c_i\rangle$ consists of $m = \lceil \log q \rceil$ qubits. Whenever we require floating point arithmetic, we follow [Her+10] assuming double precision is used, meaning $\xi = 53$ bits of precision are required. To estimate a lower bound to the cost of the minimal circuit for $\mathcal{W}$, we ignore the cost for all operations except for the bare minimum arithmetic that is required to compute the single, most expensive lattice point projection.

[Her+10] Hermans et al., "Parallel Shortest Lattice Vector Enumeration on Graphics Cards"

Components of $\mathcal{W}$

Given a tree $\mathcal{T}$ of height h, Montanaro [Mon18] defines the operator $\mathcal{W}$ using two operators R_A and R_B, the first acting on all nodes with even distance from the root, and the second on all nodes with odd distance. The implementations of the two operators are nearly identical, and as such we will

[Mon18] Montanaro, "Quantum-Walk Speedup of Backtracking Algorithms"

[Mon18] Montanaro, "Quantum-Walk Speedup of Backtracking Algorithms"

only capture the original description of R_A [Mon18, Alg. 3] by decomposing it into the following operators:

$U_{\textbf{Setup}}$: quantum operator that prepares the quantum state by advancing the variable assignment (i. e., level k of the tree), and ensures that the operator acts on the *correct* set of nodes (i. e., even or odd levels for $U_{\text{Setup}(R_A)}$ or $U_{\text{Setup}(R_A)}$, respectively).

$U_{\alpha,S}$: quantum operator that generates a superposition of children of a node, performing the map $|0\rangle \mapsto |\phi_{\alpha,S}\rangle$, where S is the set of all children of a node.

U_P: quantum operator that computes the norm of the projected lattice point being inspected, and compares the length of the projection with the pruning bound R_i. The predicate is executed to identify the children of a node.

U_0: reflection through $|0\rangle$. Together with the operator $U_{\alpha,S}$ it performs the diffusion operation $I - 2\,|\phi_{\alpha,S}\rangle\,\langle\phi_{\alpha,S}|$.

$U_{\textbf{Uncompute}}$: quantum operator to uncompute the ancillary states and the inversion of the setup step U_{Setup}.

Quantum Arithmetic

SMALLEST KNOWN ARITHMETIC CIRCUITS. The quantum arithmetic literature contains many design proposals for integer and floating point adders and multipliers. Generally, most algorithms are either "ports" of classical designs [Dra+06; Hän+20; HRS17; Nie+23; MT19], or they rely on the quantum Fourier transform to evaluate these operations [RG17; Koc+22]. As not all papers work using the same metrics or even quantum computing architectures, direct comparisons and trade-off evaluations can be difficult. For example, Pham and Svore [PS13] claim additions in constant depth and multiplications in logarithmic depth, however this seems to require a specifically designed quantum architecture. For a rule-of-thumb estimation of our attack costs, we opt to chose potentially more common asymptotics for adders and multipliers achieving the smallest T counts and depths in our literature review (other than [PS13]). We report these in Table 7.1, and ignore constants and lower order terms hidden by the O notation. Whenever numbers of different bit lengths are multiplied, we conservatively assume both to have the smaller length, since we have not found sources describing quantum circuits for unbalanced multiplication. For the "$x \leq y$" comparison operator, we use a circuit with the same asymptotic size of an adder [Dra+06]. We report the smallest (to our knowledge) in Table 7.1. During our computations, we will be ignoring constants and lower order terms hidden by the O notation.

[PS13] Pham and Svore, "A 2D nearest-neighbor quantum architecture for factoring in polylogarithmic depth."

[Dra+06] Draper et al., "A Logarithmic-Depth Quantum Carry-Lookahead Adder"

FLOATING POINT NUMBERS. We also define a value ξ that equals the amount of floating point precision required to store the coefficient of the basis vectors b_i. The literature on this topic either proposes algorithms for estimating the required precision [PS08; Det+10], appearing this to be about $\Theta(n)$, or uses double precision [Her+10]. We notice that while the `fplll` library [tea23]

[PS08] Pujol and Stehlé, "Rigorous and Efficient Short Lattice Vectors Enumeration"

[Det+10] Detrey et al., "Accelerating Lattice Reduction with FPGAs"

[Her+10] Hermans et al., "Parallel Shortest Lattice Vector Enumeration on Graphics Cards"

[tea23] team, "fplll, a lattice reduction library, Version: 5.4.2"

TABLE 7.1: Used T-gate count and depth asymptotics for quantum arithmetic circuits on x-bit numbers.

Operation	T-DEPTH	GCOST	Reference
Addition and Comparison	$O(\log x)$	$O(x)$	[Dra+06] or [Hän+20, Table 2]
Multiplication and Squaring	$O(\log^2 x)$	$O(x \log(x) \log(\log(x)))$	[Nie+23]

includes support for arbitrary precision floating point numbers, often experiments use double- or quadruple-precision floating point numbers. As a conservative choice, we observe that any setting where quantum-enumeration would be advantageous would likely result in requiring more than double-precision, since otherwise cheap classical implementations are available, and therefore consider $\xi = 53$ to be a lower bound on the required precision.

TABLE 7.2: Assumed cost of arithmetic operations to implement the predicate P. Each operation has input numbers of bit length x_i and outputs numbers of size x_{i+1}.

Operation	Input bit lengths	T-DEPTH	GCOST
1, parallel multiplication	$x_0 = \min(m, \xi)$	$\log^2(x_0)$	$h^2 \cdot x_0 \log(x_0) \log(\log(x_0))$
2, binary tree addition	$x_1 = m + \xi$	$\log h \cdot \log(x_1)$	$h^2 \cdot x_1$
3, squaring	$x_2 = x_1 + \log h$	$\log^2(x_2)$	$h \cdot x_2 \log(x_2) \log(\log(x_2))$
4, binary tree addition	$x_3 = 2\,x_2$	$\log h \cdot \log(x_3)$	$h \cdot x_3$
5, comparison	$x_4 = x_3 + \log h$	$\log(x_4)$	x_4

DEPTH AND COST ESTIMATION OF $\mathcal{W}$. In our setting, quantum enumeration is being performed on a tree $\mathcal{T}(g \in Z_k)$ corresponding to a lattice coset of dimension $h = n - k$, where n is the dimension of the full lattice $\Lambda = \Lambda(b_1, \ldots, b_n)$ being enumerated. This process would require performing arithmetic using the projected lattice basis vectors $(\pi_{n-\ell+1}(b_{n-\ell+1}), \ldots, \pi_{n-\ell+1}(b_{n-k}))$ of Λ for all levels $k < \ell \leq n$. An operator $U_P^{\min}$ is designed as to evaluate the predicate P which identifies projected vectors $v \in \pi_{n-\ell+1}(\Lambda)$ such that $||v|| \leq R_\ell$ and $\pi_{n-k+1}(v) = g$. In order to lower-bound the cost of operation, we only consider the case of evaluating this inequality at $\ell = n$, where $(\pi_{n-\ell+1}(b_{n-\ell+1}), \ldots, \pi_{n-\ell+1}(b_{n-k})) = (b_1, \ldots, b_h)$. Evaluating predicate P becomes checking whether $\left|\left|\sum_{i \leq h} c_i b_i\right|\right|^2 \leq R^2 - ||g||^2$ for some integer coefficients $(c_i)_i$. Since $(b_1, \ldots, b_h)$ span an h-dimensional vector space, we consider them to be h-dimensional by assuming an appropriate change of basis was applied.[6] Our estimate for the cost of $U_P^{\min}$ is derived as in Table 7.2 applying the following sequence of operations:

U_{Setup}: sets up the states, we assume $\text{GCOST}(U_{\text{Setup}}) = \text{T-DEPTH}(U_{\text{Setup}}) = 0$.

$U_{\alpha,S}^{\min}$: generates a superposition of the children of a node. At a bare minimum, this requires a uniform superposition $\sum_{i=0}^{q-1} |i\rangle$ which is computed by a single (parallel) layer of Hadamard gates. Since we are only accounting for T, we assume $\text{T-DEPTH}(U_{\alpha,S}^{\min}) = \text{T-DEPTH}(U_P^{\min})$ and $\text{GCOST}(U_{\alpha,S}^{\min}) = \text{GCOST}(U_P^{\min})$, as Hadamard gates do not contribute T gates.

[6] In classical implementations, this computation benefits from caching of Gram-Schmidt orthogonalisation operations and results [tea23]. Asymptotically, the number of individual arithmetic operations is the same as computing directly from $(b_1, \ldots, b_h)$.

$U_P^{\min}$: evaluates the predicate P which identifies projected vectors $v \in \pi_{n-\ell+1}(\Lambda)$ such that $||v|| \leq R_\ell$ and $\pi_{n-k+1}(v) = g$, for all levels $k < \ell \leq n$. In order to lower-bound the cost of this operation, we only consider the case of evaluating this inequality at $\ell = n$, where the condition becomes checking whether $\left|\left|\sum_{i\leq h} c_i \vec{b}_i\right|\right|^2 \leq R^2 - ||g||^2$.[7] The cost of the operation needs to account for at least the cost of the following operations (summarised in Table 7.2):

1. Parallel multiplication of h^2 pairs $(c_i, (\vec{b}_i)_j) \mapsto c_i(\vec{b}_i)_j$, of m- and ξ-bit length, outputting numbers of bit length $m + \xi$.
2. Addition of coefficients $(c_1(\vec{b}_1)_j, \ldots, c_h(\vec{b}_h)_j) \mapsto \sum_i c_i(\vec{b}_i)_j$ for $j \in [h]$. These additions can be run in parallel over j. For a fixed j, the corresponding sum is run by adding terms in pairs, forming a binary tree of sums. Each $\sum_i c_i(\vec{b}_i)_j$ output is $m + \xi + \log h$ bits long.
3. Squaring the $\sum_i c_i(\vec{b}_i)_j$ sums in parallel (output bit length $2(m + \xi + \log h)$)
4. Adding the squared sums in a binary-tree fashion to obtain $\left|\left|\sum_{i\leq h} c_i \vec{b}_i\right|\right|^2 = \sum_j (\sum_i c_i(\vec{b}_i)_j)^2$ of bit length $2(m+\xi+\log h)+\log h$.
5. The last operation is the comparison with the (adjusted) pruning bound $R^2 - ||g||^2$.

We depict the implementation of the minimal $U_P^{\min}$ implementation in Figure 7.5.

[7] In classical implementations, this computation benefits from extensive caching of Gram-Schmidt orthogonalisation operations and results [tea23]. Asymptotically, the number of individual arithmetic operations is the same as if computing directly from the basis $(b_1, \ldots, b_h)$.

U_0: the quantum operator computing $2|0\rangle\langle 0| - \mathrm{Id}$ (with Id the identity operator) which requires mult-controlled-Z gates, we estimate requiring at least one T gate.

$U_{\text{Uncompute}}$: we conservatively assume that uncomputation does not require T gates, as in the case of measurement-based uncomputation of AND gates in [Jaq+20], and assume T-DEPTH$(U_{\text{Uncompute}}) =$ GCOST$(U_{\text{Uncompute}}) = 0$.

[Jaq+20] Jaques et al., "Implementing Grover Oracles for Quantum Key Search on AES and LowMC"

Corollary 7. *The GCOST in the "Circuit-based Model" is*

$$\begin{aligned} \textit{GCOST}(\mathcal{W}) \geq\ & h^2 \cdot \min(m,\xi) \log(\min(m,\xi)) \log(\log(\min(m,\xi))) \\ & + h^2 \cdot (m+\xi) \\ & + h \cdot ((m+\xi) + \log h) \log(((m+\xi) + \log h)) \log(\log((m+\xi) + \log h)) \\ & + h \cdot 2\left((m+\xi) + \log h\right) + 2\left((m+\xi) + \log h\right) + \log h. \end{aligned} \tag{7.9}$$

Corollary 8. *The T-DEPTH in the "Circuit-based Model" is*

$$\begin{aligned} \textit{T-DEPTH}(\mathcal{W}) \geq\ & \log^2(\min(m,\xi)) + \log h \cdot \log((m+\xi)) \\ & + \log^2(((m+\xi) + \log h)) \\ & + \log h \cdot \log(2\left((m+\xi) + \log h\right)) \\ & + \log(2\left((m+\xi) + \log h\right) + \log h). \end{aligned} \tag{7.10}$$

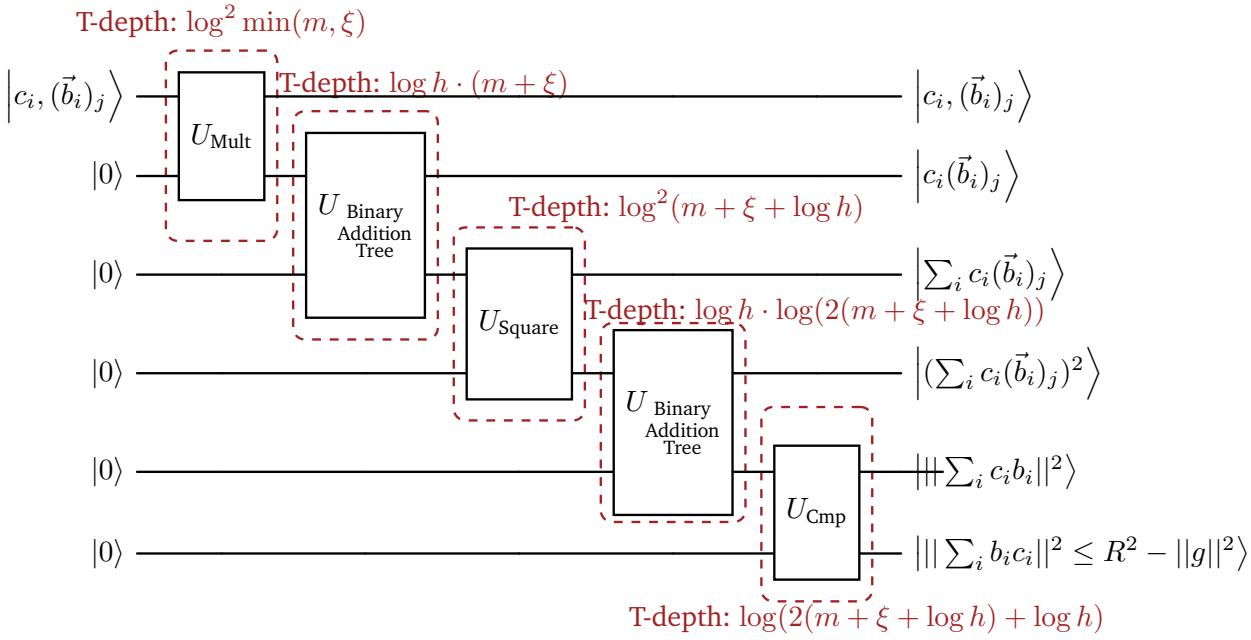


FIGURE 7.5: Minimal quantum circuit of $U_P^{\min}$.

7.3 ESTIMATING QUANTUM ENUMERATION ATTACKS ON KYBER

In Sec. 7.1 and 7.2, we have described methods to explore the enumeration tree under a MAXDEPTH limitation (cf. Sec. 7.1.4 and 7.1.5), and introduced two different instantiations of the quantum backtracking operator $\mathcal{W}$.

In this section, we leverage these results to present cost estimations for primal lattice reduction attacks using quantum enumeration against *Kyber* [Sch+22], the post-quantum KEM selected by NIST for standardization in 2022. To that end, we compute lower bounds on the cost of combined classical-quantum cylinder pruning in the gate-cost metric against the three different parametrisations of Kyber (cf. Table 7.3). For each of them, we consider attacks within MAXDEPTH $\in \{2^{40}, 2^{64}, 2^{96}\}$, as suggested by NIST [Nat17], each assuming the two different instantiations of the operator $\mathcal{W}$ outlined in Section 7.2.

[Sch+22] Schwabe et al., *CRYSTALS-KYBER*

[Nat17] National Institute for Standards and Technology, *Post-Quantum Cryptography Call for Proposals*

7.3.1 *Attack Setting*

Our aim in this section is to estimate a lower bound on the possible cost of classical-quantum enumeration in the setting of lattice-based cryptography. As a case-study, we look at the *primal attack* on Kyber, where a block lattice reduction algorithm is used to recover the secret key of the cryptosystem by reducing an embedding lattice [BG14] constructed using the public key.

[BG14] Bai and Galbraith, "Lattice decoding attacks on binary LWE"

We follow the convention of using BKZ as the lattice reduction algorithm, and assume that its SVP oracle is instantiated using our classical-quantum enumeration approach. The common approach to primal attack estimates is to choose a *cost model* for BKZ that accounts for the cost of running the SVP oracle and for the number of calls made [Alb+18]. Normally, cost models will use a closed formula for the cost of enumeration in dimension β to account for the cost of the SVP oracle, either fitted or derived from

[Alb+18] Albrecht et al., "Estimate All the LWE, NTRU Schemes!"

TABLE 7.3: Kyber parameters [Sch+22, Sec 4.3] with respective BKZ block-sizes required for the primal attack; column $\log \#\mathcal{T}^M$ reports our estimated lower bound on the number of nodes of the enumeration tree of dimension β using extreme cylinder pruning with $M = 2^{64}$ and success probability ≈ 1, following [Aon+18, Eq. (16)].

Scheme	LWE dim. n	Modulus q	Block-size β	$\log \#\mathcal{T}^M$	Targeted AES security
Kyber-512	512	3329	406	172.5	AES-128
Kyber-768	768	3329	623	323.2	AES-192
Kyber-1024	1024	3329	873	517.7	AES-256

theory or experiments. This is then used with some estimation script such as the `lwe-estimator` [APS15b], which will simulate the effect of lattice reduction and find the cheapest parametrisation of the attack leading to high success probability.

Since our setting involves an implicit relation between the gate-cost of the SVP oracle and the MAXDEPTH constraint, we do not attempt to fit our results on a curve as a function of β. Instead, we opt for calling an estimator script assuming the optimistic cost of classical enumeration obtained as part of our analysis (cf. [Bin+23, App. A]), which assumes that input bases achieve a linear lattice profile (as predicted by the Geometric Series Assumption, using the root-Hermite factor $((\pi\beta)^{1/\beta}\beta/(2\pi e))^{1/(2(\beta-1))}$ from [Che13]) resulting in a generous lower bound of the cost of solving SVP via enumeration, $2^{\beta \log \beta/8+O(\beta)}$, and assuming specifically the lower bound costs for extreme cylinder pruning proven in [Aon+18]. From this cost estimation we obtain three different block sizes β for the three parameter sets of Kyber, reported in Table 7.3. We then proceed to estimate the gate-cost of classical-quantum enumeration in dimension β under different MAXDEPTH values, and compare these with the corresponding approximate gate-cost of Grover search on AES for the corresponding category (e.g., Kyber-512 with AES-128).

It is important to highlight an issue towards claiming lower bounds on the cost of classical-quantum enumeration, and how we address it. As pointed out in [ANS18] and mentioned in Section 7.1, the expected speedup of quantum enumeration over equivalent classical enumeration may be more than quadratic, depending on the probability distribution of the size of the trees being enumerated, due to Jensen's inequality implying $\mathbb{E}[\sqrt{\#\mathcal{T}}] \leq \sqrt{\mathbb{E}[\#\mathcal{T}]}$. Since we would like to provide lower bounds to the expected attack cost, we define $z \geq 0$ such that $\mathbb{E}[\sqrt{\#\mathcal{T}}] = 2^{-z}\sqrt{\mathbb{E}[\#\mathcal{T}]}$, and estimate the attack cost for $z = 0, \ldots, 64$. While we do not know what the value of z may be for lattices encountered in cryptanalysis,[8] this allows us to delegate the estimation of the concrete cost to future analysis on the distribution of $\#\mathcal{T}$, while clearly identifying *threshold values* z_0, such that $z \geq z_0$ may imply possible effective attacks, while $z < z_0$ would indicate that classical-quantum enumeration would not threaten Kyber's security.

We note that an alternative approach could be deriving a lower bound to the Jensen's gap, depending on some other parameter of the problem. We attempt this approach in [Bin+23, App. I], where we derive bounds depending on the variance of $\#\mathcal{T}$. This, however, presents the same issue as above, namely that we are not aware of the exact distribution of $\#\mathcal{T}$. This means that while it provides a different formulation of the problem, it currently does not represent a better alternative to testing many values of

[APS15b] Albrecht, Player, and Scott, "On the concrete hardness of Learning with Errors"

[Bin+23] Bindel et al., *Quantum Lattice Enumeration in Limited Depth*

[Che13] Chen, "Reduction de reseau et securite concrete du chiffrement completement homomorphe"

[Aon+18] Aono et al., "Lower Bounds on Lattice Enumeration with Extreme Pruning"

[ANS18] Aono, Nguyen, and Shen, "Quantum Lattice Enumeration and Tweaking Discrete Pruning"

[8] Albeit not at cryptographically relevant sizes, in [Bin+23, App. D] we present the results of small-dimension measurements of the Jensen's gap against q-ary lattices. For pruned enumeration in dimension $\beta \approx 60$, the gap appears to be around $z \approx 1$.

z and looking for threshold values. We do, however, note that preliminary results in [Bin+23, App. E] suggest that the distribution may be relatively narrowly distributed about its mean.

[Bin+23] Bindel et al., *Quantum Lattice Enumeration in Limited Depth*

Overall, our estimation code for the cost of enumeration on a β-dimensional lattice bases is given on input a multiplicative Jensen's gap 2^z, a MaxDepth constraint, a number of bases M used during extreme pruning (here, $M = 2^{64}$), a maximum number Y of tree nodes that can be stored in QRACM (here, $Y = 2^{64}$), an estimate on the size of the quantum backtracking operator $\mathcal{W}$ and on the values for DF, QD, and WQ, and pruning parameters for estimating upper bounds on H_k, and optimizes the level k which separates classical and quantum enumeration as well as the maximal number 2^y of nodes on this level to be combined under a virtual root, looking for the cheapest possible attack. We estimate the cost assuming extreme pruning attacks targeting success probability ≈ 1.

An overview over all parameters used in the estimation process is given in Table 7.4, while the costing loop is presented in Figure 7.6. We have made the source code used to produce our experimental results, tables, and plots publicly available.

TABLE 7.4: Attack parameters for experimental evaluation.

$\text{MaxDepth} \in \{2^{40}, 2^{64}, 2^{96}\}$,	$y \in \{0, \ldots, 64\}$,	$z \in \{0, \ldots, 64\}$,	$M = 2^{64}$,	
$n \in \{406, 623, 873\}$,	$k \in [n]$,	$h = n-k+1$,	$\text{DF}(\cdot) = \text{QD}(\cdot) = 1$,	$\text{WQ}(\mathcal{T}) = \sqrt{\#\mathcal{T} \cdot h}$

$\text{CostingLoop}(n, \text{MaxDepth}, Y = 2^{64}, Z = 2^{64})$

```
for z ∈ {0, 1, 2, ..., log(Z)}
    k ← n + 1
    while LB(T-Depth(QPE(W))) ≤ MaxDepth and k ≥ 0
        y ← Largest y ∈ {0, 1, 2, ..., log(Y)}
                s.t. LB(T-Depth(QPE(W))) ≤ MaxDepth
        // store classical cost and quantum cost
        CC ← E_{random tree T} [Classical GCost]
        QC ← LB( E_{random tree T} [Quantum GCost])
        GCost[z][y, k] ← CC + QC
        k ← k − 1
return ( min_{y,k} (GCost[z]) )_{z∈{0,1,2,...,log(Z)}}
```

FIGURE 7.6: Pseudocode for cost estimation under MaxDepth constraint, following Equation (7.8). $\text{GCost}[z][y, k]$ is the total cost associated with the quantum enumeration (i. e., the sum of Equation (7.7) and Equation (7.5)) with $M = 2^{64}$. Operator $\mathcal{W}$ is instantiated according to Section 7.2.1 and Section 7.2.2, respectively. $LB(\cdot)$ stands for "lower bound of".

7.3.2 *Cost Estimation of the Attack*

COST METRICS AND SUCCESS CONDITIONS. As mentioned in Section 3.3, the cost of a quantum algorithm can be measured using various metrics. In this paper, we prefer to focus on the number of classical and quantum gates required by the attack in total, since, plausibly, applying one quantum gate requires running one classical computation on some microcontroller [JS19],

[JS19] Jaques and Schanck, "Quantum Cryptanalysis in the RAM Model: Claw-Finding Attacks on SIKE"

meaning that to some extent these two quantities can be compared and combined. As such, one could say a classical-quantum enumeration attack was successful if the total number of gates required[9] was lower than some threshold capturing some security notion.

The success of an attack can be defined in multiple ways. One can note that submissions to the NIST standardization process, such as Kyber, were required to propose parameters for cryptographic primitives as hard to break as AES or SHA (depending on the targeted security category). This would imply a notion where a quantum attack against Kyber may be considered successful only if its cost is lower than the number of gates required to run Grover search against AES, which we estimate using Tables 10 and 12 of [Jaq+20].[10] It should be noticed that the reason for such a separation between classical and quantum attacks is due to the assumed and yet-unknown hidden costs of quantum computation.

While the comparison with the cost of Grover on AES is our primary success metric for the attack, in [Bin+23, App. G.1] we investigate the cost of the attack with respect to possible alternative success metrics. In the following paragraphs we proceed to explore whether combined classical-quantum enumeration could *plausibly* be cheaper than Grover search on AES, where plausibility depends on the value of the multiplicative Jensen's gap required (cf. Definition 7.1.1). To be conservative, we compare the cost of Grover search to a simple sum of classical and quantum gate costs for enumeration, which is likely an extremely generous approach towards quantum computation cost estimation.

DEPENDENCE ON PRUNING PARAMETERS. Since enumeration is being performed on pruned enumeration trees, pruning parameters play an important role in determining the cost of the attacks. Lower bounds for the pruning radii and for the values of H_k in the extreme cylinder pruning setting can be found in [Aon+18], while upper bounds can be found via optimization techniques, thanks to the seminal work of Gama, Nguyen and Regev [GNR10]. We discuss in more detail how we obtain both bounds in [Bin+23, App. A.2]. Yet, we want to briefly speculate about three different ways these can be combined to produce different attack cost estimates that we reproduce in Table 7.5.

Subtree-size estimation is performed leveraging Conjecture 3 where the ratios H_{k+j}/H_k are used to lower bound $\mathbb{E}[|Z_{k+j}|/|Z_k|]$. A first cautious approach, that we label "**LB/UB**", is to strictly lower bound H_{k+j}/H_k by taking the lower bound of H_{k+j} and dividing it by the upper bound of H_k. A more speculative approach could be that of assuming that the pruning parameters obtained via optimisation, and used to determine the upper bound, cannot be significantly improved. In this scenario, that we label "**UB/UB**", the upper bounds are assumed to be exact values, and the previous numerator can be replaced with an upper bound for H_{k+j}. Finally, one could instead speculate that the optimal pruning parameters found are only a local optimum, and pruning radii closer to the lower bounds can potentially be found. In this scenario, that we label "**LB/LB**", one could imagine that the radii for H_k, when $k < n$, could be improved, leading to $\mathbb{E}[|Z_{k+j}|/|Z_k|]$ being closer to the ratio of lower bounds from [Aon+18]. This latter scenario

[9] We lower bound this as the number of nodes visited in the classical phase plus the number of quantum gates applied during the quantum phase of the attack.

[Jaq+20] Jaques et al., "Implementing Grover Oracles for Quantum Key Search on AES and LowMC"

[10] Under no MAXDEPTH constraint, [Jaq+20, Table 10] suggests that the GCOST of key recovery against AES-128 (resp. AES-192, AES-256) with success probability ≈ 1 is $\approx 2^{83}$ (resp. 2^{115}, 2^{148}). Under a depth constraint, [Jaq+20, Table 12] suggests the GCOST $\approx 2^{157}/\text{MAXDEPTH}$ (resp. $2^{221}/\text{MAXDEPTH}$, $2^{285}/\text{MAXDEPTH}$). We note that further improvements in the design of the Grover oracles against AES have achieved minor speedups in terms of overall gate cost.

[Bin+23] Bindel et al., *Quantum Lattice Enumeration in Limited Depth*

[Aon+18] Aono et al., "Lower Bounds on Lattice Enumeration with Extreme Pruning"

[GNR10] Gama, Nguyen, and Regev, "Lattice Enumeration Using Extreme Pruning"

could counter-intuitively reduce the overall cost of classical enumeration, but increase the average size of subtrees rooted on level k. In the remainder of the article, we report gate-cost estimates in these three scenarios.

Cost Estimation without MaxDepth Restrictions.

We start by estimating the cost of enumeration without MaxDepth restrictions—the most favorable setting to the adversary. In this setting, a quadratic speedup in terms of quantum depth can be achieved as the full enumeration tree can be enumerated directly within a call to FindMV, meaning no classical phase is required. This means, the attack consists of calling FindMV($\mathcal{T}^M$) once. No QRACM is needed, and the classical cost is null. We do notice that this is not necessarily optimal but we consider it as it is a good reference for the cost of other attacks. We report the cost of this attack under MD $= \infty_{k=0}$ in Table 7.5. This is also the state of the art prior to the introduction of combined classical-quantum enumeration. The dependency between the cost of the attack and the Jensen's gap 2^z is straightforward in this setting, with the quantum cost exponentially reducing as z increases. It appears from our estimates that quantum enumeration on Kyber-768 and -1024 may be more expensive than key-search on AES in this setting. Only Kyber-512 appears to be plausibly approachable, $z \geq 7$ sufficing, yet this is still only considering the query model for $\mathcal{W}$. A significantly larger Jensen's gap of $z \geq 32$ is already required in the circuit-based model of $\mathcal{W}$ in Section 7.2.2. We remark that in this setting Grover search on AES can be very competitive, achieving a full quadratic speedup.

We also consider the cost of running our combined classical-quantum enumeration attack in unlimited depth. Differently from the attack mentioned above, we decide not to fit the entire tree within one quantum enumeration, and rather first perform an optimal amount of classical precomputation. We report the results of this cost estimation in Table 7.5, under the "MD $= \infty$" rows and show how the quantum and classical cost perform in Figure 7.7. This is the minimal cost we find when unlimited quantum depth is available. The results are quite similar to those of fully-quantum enumeration, with only Kyber-512 and -768 appearing possibly easier to attack in the case where Conjecture 3 is instantiated using "**LB/UB**" numbers. Yet, even the most aggressive setting ($\mathcal{W}$ as in Section 7.2.1, using **LB/UB**), the query complexity of quantum enumeration on Kyber-512 essentially matches the query complexity of Grover search on AES-128 in the same unbounded depth setting, with the query-complexity of enumeration on Kyber-768 is greater than that of Grover search on AES-192.

Cost estimation with MaxDepth restrictions.

We now consider the effect of depth restrictions on the cost of the attack. Depth restrictions mean that we will need to use a combined classical-quantum attack as described in Sec. 7.1.3 and 7.1.4, where classical enumeration is run up to level k, as to create subsets $\{g_1, \ldots, g_{2^y}\} \subset Z_k$. A "virtual" root node v is added as "parent" of these, and quantum enumeration is run on the resulting tree $\mathcal{T}(v)$. This process requires about 2^y QRACM to store

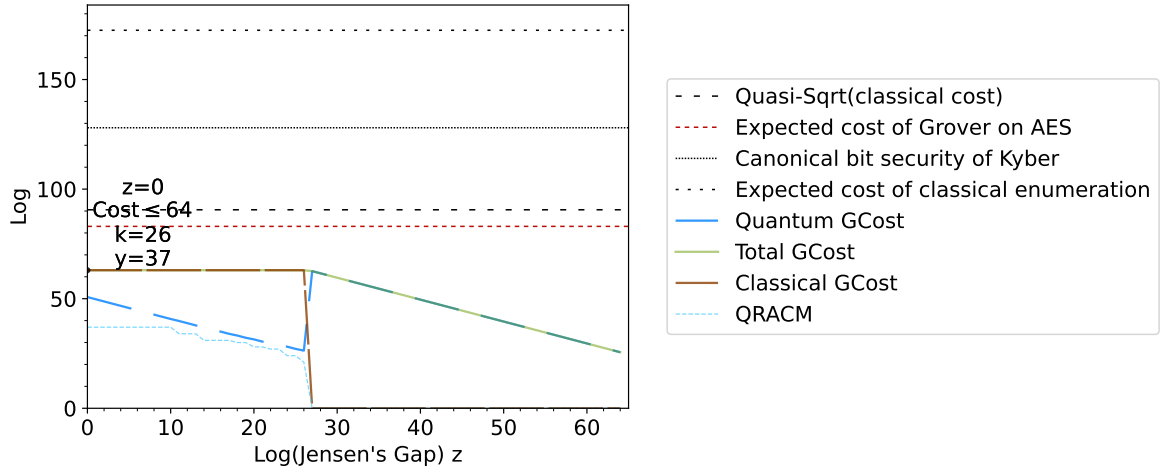
TABLE 7.5: Summary of the values for the Jensen's gap 2^z at crossover points of our combined classical-quantum enumeration attacks against Kyber and the cost of Grover's search against AES (cf. [Jaq+20, Tables 10 and 12]). We remark that exact crossovers happen at fractional values of z. In this table we round down threshold values of z. MAXDEPTH is abbreviated to MD. X/Y refers to how $\mathbb{E}[|Z_{k+j}|/|Z_k|]$ is estimated for displayed level k (c.f. Section 7.3.2), Cost is as in Table 7.6.

less likely to be feasible — more likely to be feasible

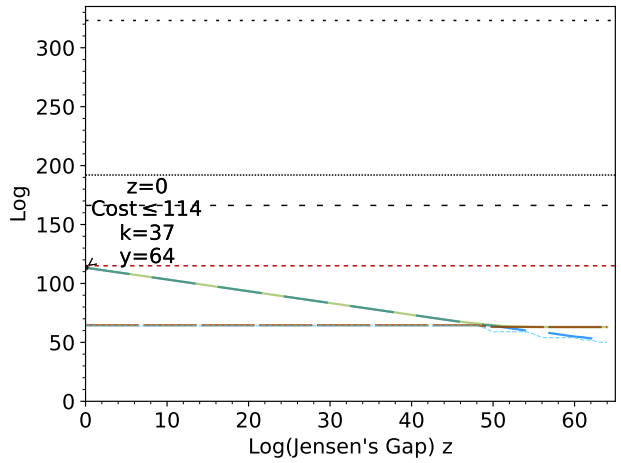
		Crossover points when comparing Cost of Grover on AES against the total GCOST (cf. Table 7.6) with …					
		…$\mathcal{W}$ as in Section 7.2.1			…$\mathcal{W}$ as in Section 7.2.2		
MD	Kyber	LB/UB	UB/UB	LB/LB	LB/UB	UB/UB	LB/LB
2^{40}	-512	$z \geq 0, k \leq 25$ Cost $\geq 2^{63}$	$z \geq 20, k \leq 11$ Cost $\geq 2^{116}$	$z \geq 12, k \leq 83$ Cost $\geq 2^{115}$	$z \geq 0, k \leq 27$ Cost $\geq 2^{94}$	$z \geq 36, k \leq 22$ Cost $\geq 2^{115}$	$z \geq 28, k \leq 79$ Cost $\geq 2^{115}$
	-768	$z \geq 2, k \leq 84$ Cost $\geq 2^{179}$	$z \geq 61, k \leq 33$ Cost $\geq 2^{180}$	$z \geq 56, k \leq 106$ Cost $\geq 2^{179}$	$z \geq 17, k \leq 73$ Cost $\geq 2^{180}$	$z > 64$	$z > 64$
	-1024	$z \geq 50, k \leq 105$ Cost $\geq 2^{242}$	$z > 64$	$z > 64$	$z > 64$	$z > 64$	$z > 64$
2^{64}	-512	$z \geq 0, k \leq 26$ Cost $\geq 2^{63}$	$z \geq 20, k \leq 9$ Cost $\geq 2^{92}$	$z \geq 12, k \leq 64$ Cost $\geq 2^{91}$	$z \geq 0, k \leq 26$ Cost $\geq 2^{75}$	$z \geq 36, k \leq 5$ Cost $\geq 2^{92}$	$z \geq 28, k \leq 54$ Cost $\geq 2^{91}$
	-768	$z \geq 1, k \leq 64$ Cost $\geq 2^{155}$	$z \geq 61, k \leq 26$ Cost $\geq 2^{156}$	$z \geq 56, k \leq 77$ Cost $\geq 2^{155}$	$z \geq 17, k \leq 67$ Cost $\geq 2^{156}$	$z > 64$	$z > 64$
	-1024	$z \geq 49, k \leq 100$ Cost $\geq 2^{220}$	$z > 64$	$z > 64$	$z > 64$	$z > 64$	$z > 64$
2^{96}	-512	$z \geq 0, k \leq 26$ Cost $\geq 2^{63}$	$z \geq 15, k \leq 1$ Cost $\geq 2^{82}$	$z \geq 7, k \leq 40$ Cost $\geq 2^{82}$	$z \geq 0, k \leq 26$ Cost $\geq 2^{75}$	$z \geq 40, k \leq 1$ Cost $\geq 2^{82}$	$z \geq 31, k \leq 40$ Cost $\geq 2^{82}$
	-768	$z \geq 1, k \leq 53$ Cost $\geq 2^{124}$	$z \geq 61, k \leq 8$ Cost $\geq 2^{124}$	$z \geq 56, k \leq 44$ Cost $\geq 2^{123}$	$z \geq 19, k \leq 43$ Cost $\geq 2^{124}$	$z > 64$	$z > 64$
	-1024	$z \geq 51, k \leq 79$ Cost $\geq 2^{187}$	$z > 64$	$z > 64$	$z > 64$	$z > 64$	$z > 64$
∞	-512	$z \geq 0, k \leq 26$ Cost $\geq 2^{63}$	$z \geq 15, k \leq 1$ Cost $\geq 2^{82}$	$z \geq 7, k \leq 40$ Cost $\geq 2^{82}$	$z \geq 0, k \leq 26$ Cost $\geq 2^{75}$	$z \geq 40, k \leq 1$ Cost $\geq 2^{82}$	$z \geq 31, k \leq 40$ Cost $\geq 2^{82}$
	-768	$z \geq 0, k \leq 37$ Cost $\geq 2^{113}$	$z \geq 59, k \leq 3$ Cost $\geq 2^{114}$	$z \geq 51, k \leq 31$ Cost $\geq 2^{114}$	$z \geq 24, k \leq 37$ Cost $\geq 2^{114}$	$z > 64$	$z > 64$
	-1024	$z \geq 59, k \leq 33$ Cost $\geq 2^{147}$	$z > 64$	$z > 64$	$z > 64$	$z > 64$	$z > 64$
$\infty_{k=0}$	-512	$z \geq 7, k = 0$ Cost $\geq 2^{82}$	$z \geq 15, k = 0$ Cost $\geq 2^{82}$	$z \geq 7, k = 0$ Cost $\geq 2^{82}$	$z \geq 32, k = 0$ Cost $\geq 2^{82}$	$z \geq 40, k = 0$ Cost $\geq 2^{82}$	$z \geq 32, k = 0$ Cost $\geq 2^{82}$
	-768	$z \geq 51, k = 0$ Cost $\geq 2^{114}$	$z \geq 56, k = 0$ Cost $\geq 2^{114}$	$z \geq 51, k = 0$ Cost $\geq 2^{114}$	$z > 64$	$z > 64$	$z > 64$
	-1024	$z > 64$	$z > 64$	$z > 64$	$z > 64$	$z > 64$	$z > 64$

the $\{g_i\}_{i \leq 2^y}$, and is run on the extreme cylinder pruning enumeration tree $\mathcal{T}^M$ from Section 7.1.5.

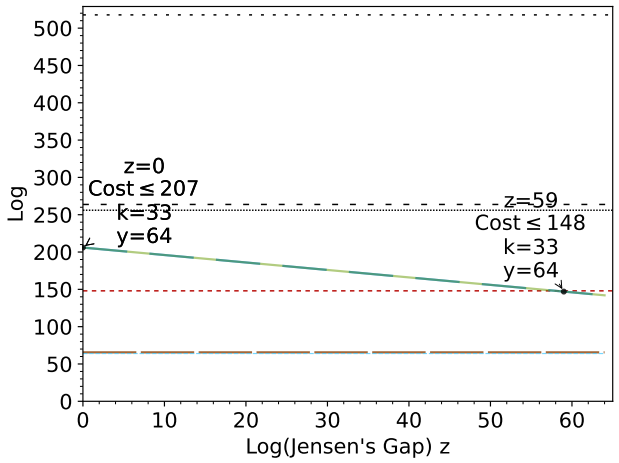
To compute Equation (7.7) given a Jensen's gap 2^z, we minimize the total cost of the combined classical-quantum enumeration with extreme cylinder pruning using $M = 2^{64}$ re-randomized bases and overall success probability ≈ 1 over $2^y \leq 2^{64}$ and $k \leq n$. For each set of parameters, we only consider those where the depth of QPE($\mathcal{W}$) (cf. Equation (7.8)) is no larger than MAXDEPTH. For every $z \in \{0, ..., 64\}$, we output (y, k) minimizing the



(a) Cost estimation for Kyber-512.



(b) Cost estimation for Kyber-768.



(c) Cost estimation for Kyber-1024.

FIGURE 7.7: Cost estimation for Kyber without MAXDEPTH restrictions with the instantiation for operator $\mathcal{W}$ as in Section 7.2.1 and with DF $= 1$, QD $= 1$, $b = 1$ (see Section 7.1), corresponding to the lower bound (LB/UB) for the Conjecture 3.

total cost. We report our results in Table 7.5, under the "MD $= 2^{40}, 2^{64}, 2^{96}$" rows. First, we observe that attacks on Kyber-1024 appear unlikely to beat key-search on AES-256 in all settings. The value of z required to reach an attack costs on Kyber-512 and -768 smaller than those of Grover on AES is relatively low when assuming the strict "LB/UB + query-based model for $\mathcal{W}$" setting (cf. Section 7.2.1). However, assuming the circuit-based model for $\mathcal{W}$, immediately raises the requirements for a successful attack on Kyber-768 up to $z \geq 17$, suggesting a successful attack may not be too likely. This is likely a fairer comparison, since in the "$\mathcal{W}$ as in Section 7.2.1" columns we are anyway comparing query-complexity of enumeration versus gate-complexity of Grover on AES. As for Kyber-512, while we cannot fully exclude attacks due to the very conservative analysis made, we note that the estimated gate cost of the attack significantly increases assuming the need for non-trivial circuits for $\mathcal{W}$ ($2^{63} \rightarrow 2^{75}$–$2^{94}$), or that the pruning radii found via optimisation cannot be improved (2^{82}–2^{116}, c.f. the "UB/UB" columns). We note that the cost of Kyber-512 is the same in the MAXDEPTH $= 2^{96}$ and MAXDEPTH $= \infty$

cases, since in either case the quantum depth budget is large enough to fit the attack achieving the overall optimal classical-quantum enumeration tradeoff.

We remark that for all attacks identified within $z \leq 64$, we have $k \leq n/2$. This matches our analysis in Section 7.1.4 since as $k \to n/2$, the cost of the classical phase of combined classical-quantum enumeration would approach the cost of fully classical enumeration, while introducing a further quantum overhead.

TABLE 7.6: Legend for plots and tables reporting attack costs under MAXDEPTH constraint.

Exp. cost of Grover on AES	Expected GCOST for AES key recovery [Jaq+20, Tab. 12] with prob. ≈ 1
Quantum GCOST	Expected combined cost of all quantum circuits enumerating levels below k, cf. Equation (7.7)
Exp. cost of class. enum.	Lower bounds on the cost of enumeration with extreme cylinder pruning [Aon+18]
Classical GCOST	Expected # nodes (cf. Equation (7.5)) enumerated classically up to level k
2^{128}, 2^{192}, 2^{256}	Canonical bit security
Total GCOST	Classical cost + quantum cost
Quasi-Sqrt (class. cost)	Asymptotic runtime of quantum enumeration, $\approx (2^y \cdot N^M_{k,n-k} \cdot (n-k))^{1/2}$
QRACM	Max. amount of quantum accessible classical memory, constraint on 2^y
2^z	Multiplicative Jensen's Gap, cf. Def. 7.1.1
Cost	Value of total GCOST at this point
k	Level up to which tree is enumerated classically
MAXDEPTH	Constraint on T-DEPTH(QPE($\mathcal{W}$)), cf. Equation (7.8)
2^y	# subtrees rooted at level k combined under a single FINDMV call, cf. Sec. 7.1.4

An important difference between the bounded- and unbounded-depth settings for combined classical-quantum enumeration is the dependency of the total cost on the Jensen's gap 2^z. Indeed, while in the unbounded setting the cost of the attack is simply proportional to 2^{-z}, in the bounded setting different values of MAXDEPTH and z imply different amounts of classical precomputation.

Since we do not have a clear prediction of the exact value of z for different enumeration tree distributions, we investigate how sensitive the total cost of the attack is to small changes in z by plotting the predicted classical and quantum gate costs and QRACM requirements as a function of z. In Figure 7.8 we show the resulting plots for Kyber-1024 at MAXDEPTH $= 2^{40}$ and 2^{96} in the query-based model as a representative example. Plots for MAXDEPTH $= 2^{64}$, Kyber-512 and -768, and for the circuit-based model can be found in [Bin+23, App. G]. Overall, costs appear to decrease smoothly as z increases without major sudden changes. A peculiar phenomenon can be observed, namely the optimal attack is not achieved when the two phases of the attack are balanced. We will elaborate on this next.

[Bin+23] Bindel et al., *Quantum Lattice Enumeration in Limited Depth*

UNBALANCED CLASSICAL AND QUANTUM COST. In Figure 7.8 as well as in [Bin+23, App. G], the figures show a gap between the costs of the classical and quantum phases, rather than having these be balanced. The only parameter determining the classical cost is k. Since in our observed case the classical cost is always smaller than the quantum one, the only option for balancing the two would be by increasing k. Increasing $k \mapsto k+1$ means that the classical cost increases by an additive term $H_{k+1}/2$, while the quantum cost and the quantum depth approximately change by a factor $\sqrt{\frac{n-k-1}{n-k}\frac{H_{k+1}}{H_k}}$ and $\sqrt{\frac{n-k-1}{n-k}\frac{H_k}{H_{k+1}}}$, respectively. How this affects the quantum cost overall will depend on whether H_{k+1}/H_k is larger or smaller than 1, as well as whether a different value of y is chosen as to keep using exactly MAXDEPTH quantum depth during the attack. From Table 7.5, it appears that the optimal attacks we find are in the $k < n/2$ regime where $H_{k+1}/H_k > 1$ [GNR10], meaning that increasing k may increase both classical and quantum costs, which is undesirable. Due to the complexity of an analytic analysis, we believe the safer approach is looking for the optimal attack computationally. It would then appear that the lowest *classical plus quantum* cost is achieved with unbalanced quantum and classical costs within the constraints we consider.

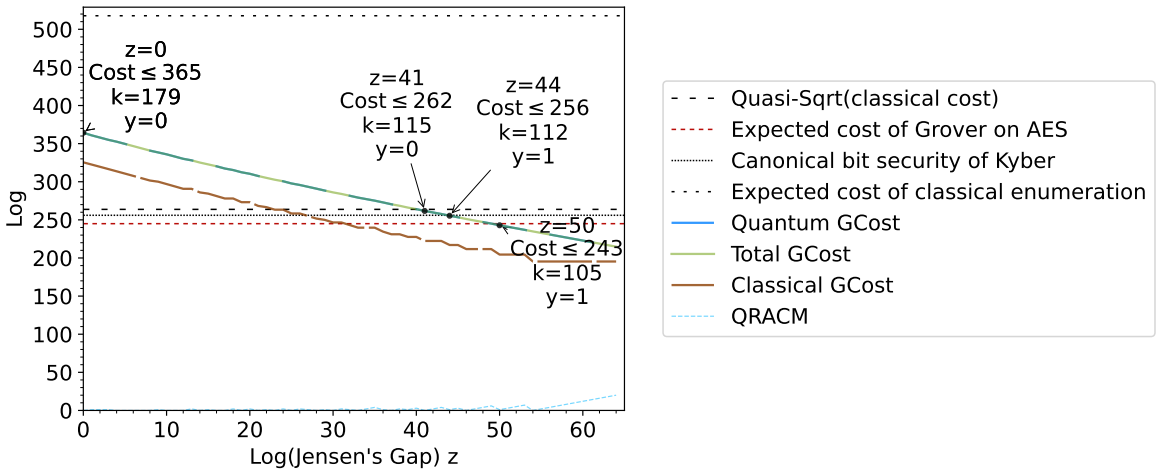
[Bin+23] Bindel et al., *Quantum Lattice Enumeration in Limited Depth*

[GNR10] Gama, Nguyen, and Regev, "Lattice Enumeration Using Extreme Pruning"

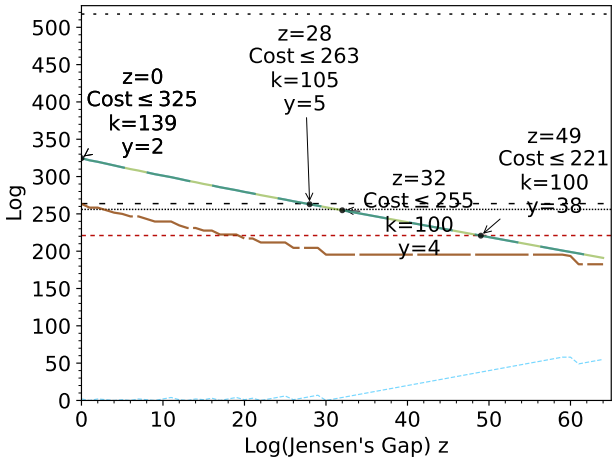
Experimental Results beyond lower bounds for QD$(\mathcal{W})$ and WQ$(\mathcal{T},\mathcal{W})$.

We explore more likely values for QD$(\mathcal{W})$ and WQ$(\mathcal{T},\mathcal{W})$ in Section 7.1.2 and re-estimate the costs presented in this section, resulting in Table 7.7. Overall, comparing the results Table 7.7 with Table 7.5 it would appear that the impact of using more likely values over the query-model numbers in Table 7.5 is smaller than the impact of moving from a query-based to a circuit-based model for $\mathcal{W}$.

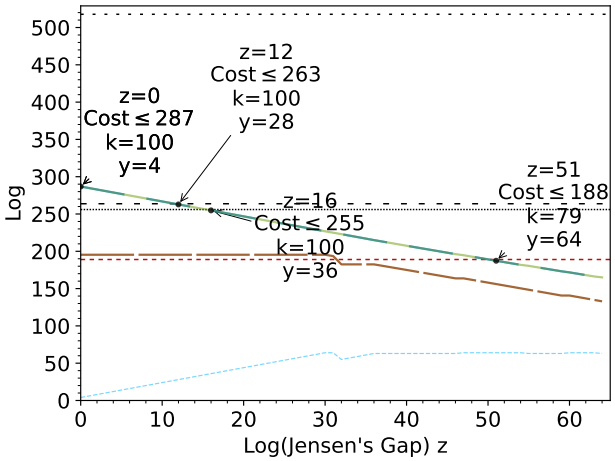
Further, we replicate the analysis performed in Section 7.3.2, estimating the cost of a combined classical-quantum attack as described in Sec. 7.1.3 and 7.1.4. In Tables B.3 and B.4 we summarize the values for the Jensen's gap 2^z at crossover points of our combined classical-quantum enumeration attacks against the quasi-square-root and the canonical $128, 192, 256$ bit security of Kyber. The tables corresponds to attacks in the settings for $\mathcal{W}$ as in Section 7.2.1 and Section 7.2.2 with DF and QD as in Section 7.1.2, $\mathcal{C} = 2$ (as this is the most conservative value such that $n \log \mathcal{C} > 0$) $b = 1/64$ in WQ$(\mathcal{T}, W)$ and $\varepsilon = 20$.



(a) Kyber-1024, MaxDepth $= 2^{40}$



(b) Kyber-1024, MaxDepth $= 2^{64}$



(c) Kyber-1024, MaxDepth $= 2^{96}$

FIGURE 7.8: Cost estimation for Kyber-1024 under MaxDepthrestrictions with the instantiation for operator $\mathcal{W}$ as in Section 7.2.1 corresponding to the lower bound (LB/UB) for Conjecture 3, cf. Table 7.6 for an expanded legend.

TABLE 7.7: Summary of the values for the Jensen's gap 2^z at crossover points of our combined classical-quantum enumeration attacks against Kyber and the cost of Grover's search against AES (cf. [Jaq+20, Tables 10 and 12]). We remark that exact crossovers happen at fractional values of z. In this table we round down threshold values of z. MAXDEPTH is abbreviated to MD. Cost is as in Table 7.6. The results are estimated from the bounds $\mathcal{C} = 2$, $\varepsilon = 20$, $b = 1/64$.

less likely to be feasible — more likely to be feasible

Crossover points when comparing Grover on AES against $\log \mathbb{E}[\text{Quantum GCOST}]$ (cf. Equation (7.7)) with …

		…$\mathcal{W}$ as in Section 7.2.1			…$\mathcal{W}$ as in Section 7.2.2		
MD	Kyber	LB/UB	UB/UB	LB/LB	LB/UB	UB/UB	LB/LB
2^{40}	-512	$z \geq 0, k \leq 25$ Cost $\geq 2^{90}$	$z \geq 34, k \leq 11$ Cost $\geq 2^{116}$	$z \geq 26, k \leq 83$ Cost $\geq 2^{115}$	$z \geq 4, k \leq 31$ Cost $\geq 2^{115}$	$z \geq 50, k \leq 19$ Cost $\geq 2^{115}$	$z \geq 41, k \leq 79$ Cost $\geq 2^{116}$
	-768	$z \geq 15, k \leq 75$ Cost $\geq 2^{180}$	$z > 64$	$z > 64$	$z \geq 32, k \leq 67$ Cost $\geq 2^{179}$	$z > 64$	$z > 64$
	-1024	$z \geq 64, k \leq 100$ Cost $\geq 2^{244}$	$z > 64$	$z > 64$	$z > 64$	$z > 64$	$z > 64$
2^{64}	-512	$z \geq 0, k \leq 26$ Cost $\geq 2^{72}$	$z \geq 34, k \leq 4$ Cost $\geq 2^{92}$	$z \geq 26, k \leq 55$ Cost $\geq 2^{91}$	$z \geq 5, k \leq 26$ Cost $\geq 2^{92}$	$z \geq 52, k \leq 1$ Cost $\geq 2^{92}$	$z \geq 43, k \leq 40$ Cost $\geq 2^{92}$
	-768	$z \geq 15, k \leq 64$ Cost $\geq 2^{156}$	$z > 64$	$z > 64$	$z \geq 32, k \leq 67$ Cost $\geq 2^{155}$	$z > 64$	$z > 64$
	-1024	$z \geq 64, k \leq 100$ Cost $\geq 2^{220}$	$z > 64$	$z > 64$	$z > 64$	$z > 64$	$z > 64$
2^{96}	-512	$z \geq 0, k \leq 26$ Cost $\geq 2^{72}$	$z \geq 37, k \leq 1$ Cost $\geq 2^{82}$	$z \geq 29, k \leq 40$ Cost $\geq 2^{82}$	$z \geq 15, k \leq 26$ Cost $\geq 2^{82}$	$z \geq 62, k \leq 1$ Cost $\geq 2^{82}$	$z \geq 53, k \leq 40$ Cost $\geq 2^{82}$
	-768	$z \geq 17, k \leq 44$ Cost $\geq 2^{123}$	$z > 64$	$z > 64$	$z \geq 37, k \leq 37$ Cost $\geq 2^{124}$	$z > 64$	$z > 64$
	-1024	$z > 64$	$z > 64$	$z > 64$	$z > 64$	$z > 64$	$z > 64$
∞	-512	$z \geq 0, k \leq 26$ Cost $\geq 2^{72}$	$z \geq 37, k \leq 1$ Cost $\geq 2^{82}$	$z \geq 29, k \leq 40$ Cost $\geq 2^{82}$	$z \geq 15, k \leq 26$ Cost $\geq 2^{82}$	$z \geq 62, k \leq 1$ Cost $\geq 2^{82}$	$z \geq 53, k \leq 40$ Cost $\geq 2^{82}$
	-768	$z \geq 22, k \leq 37$ Cost $\geq 2^{114}$	$z > 64$	$z > 64$	$z \geq 47, k \leq 37$ Cost $\geq 2^{114}$	$z > 64$	$z > 64$
	-1024	$z > 64$	$z > 64$	$z > 64$	$z > 64$	$z > 64$	$z > 64$
$\infty_{k=0}$	-512	$z \geq 29, k = 0$ Cost $\geq 2^{82}$	$z \geq 37, k = 0$ Cost $\geq 2^{82}$	$z \geq 29, k = 0$ Cost $\geq 2^{82}$	$z \geq 54, k = 0$ Cost $\geq 2^{82}$	$z \geq 62, k = 0$ Cost $\geq 2^{82}$	$z \geq 54, k = 0$ Cost $\geq 2^{82}$
	-768	$z > 64$	$z > 64$	$z > 64$	$z > 64$	$z > 64$	$z > 64$
	-1024	$z > 64$	$z > 64$	$z > 64$	$z > 64$	$z > 64$	$z > 64$

Conclusion

We have demonstrated significant vulnerabilities and attack vectors within candidates of the NIST post-quantum competition, advancing both the theoretical and practical understanding of their security and associated adversarial costs.

Firstly, we have presented an attack that exploits decryption failures in Mersenne number cryptosystems, effectively compromising IND-CCA security and even enabling the extraction of the secret-key. In particular, we showed that failing ciphertexts can be used to estimate the bit positions of the ones in the secret. We presented an attack leveraging those failures and used our estimation to break the IND-CCA security of the Ramstake cryptosystem. The analysis is based on two heuristic arguments which were supported by empirical evaluations and allowed to derive a maximum likelihood estimator for the private key. Based on the estimator we derive information about the secret to perform a Slice-and-Dice attack with significantly reduced complexity. Our implementation demonstrates the feasibility of the attack and shows that we can reconstruct the secret-key with approximately 2^{74} queries to the decryption oracle and about 2^{46} iterations of Grover's algorithm.

Secondly, our research has identified the optimal positions for targeting SPHINCS$^+$ with generic quantum preimage attacks, highlighting critical points of interest in the hypertree. We proposed and reviewed multiple points of attack in the SPHINCS$^+$ signature scheme based on applying Grover's algorithm to find preimages. An estimate of the resources required to carry out the most promising attack on a fault tolerant quantum computer is given. Our attack, that forges a signature with $1.55 \cdot 2^{101}$ logical-qubit-sycles, improves over the previously best known attack on SPHINCS$^+$-128-Haraka. Following the suggestion by NIST to review the security in terms of a maximal depth for quantum circuits, it is clear that for a depth of 2^{96} the attack can be implemented without any further constraints and would be more efficient than the classical counter part. For a depth of 2^{40} and 2^{64} the overhead induced by error correction needs to be reevaluated and optimized to the respective depth.

Lastly, we have explored and established meaningful lower bounds for quantum lattice enumerations, providing valuable insights into the computational efforts required by adversaries. We introduced a new quantum algorithm that combines classical and quantum enumeration to circumvent likely restrictions to serial quantum computation, developed a heuristic analysis of its cost in terms of classical and quantum gates and quantum depth, provided lower bounds for the cost, and studied its hypothetical impact on the cryptanalysis of Kyber in various settings as a case-study. On the way, we produced various experimental results on the distribution of subtrees of enumeration trees, and on the hidden constants of quantum enumeration algorithms. From our estimates on Kyber, we see that the asymptotic square-root speedup suggested by previous analysis of quantum enumeration with extreme cylinder pruning are not necessarily guaranteed under a MaxDepth constraint. Rather, achieving asymptotic speedups and "breaks" depends

on a vast array of hypothetical developments, such as cheap quantum computation and QRACM, better pruning radii and small quantum circuits for floating point arithmetic, and on known unknowns such as the Jensen's gap for the distribution of enumeration subtree sizes. While we can say with some confidence that quantum enumeration does not seem to threat parameters in the Kyber-1024 regime, the picture is less clear for smaller schemes. Yet, we stress again the very conservative nature of our analysis. Requiring non-trivial circuits for $\mathcal{W}$ such as those in [Bai+23] would likely imply security with respect to AES for Kyber-768 and large absolute gate-costs for attacks against Kyber-512. We believe that the take-home message of this case-study is that, as analogously noticed in the key-search setting [Jaq+20], imposing MaxDepth limitations to quantum backtracking appears to present a significant obstacle towards leveraging this technique for lattice cryptanalysis.

[Bai+23] Bai et al., "Concrete Analysis of Quantum Lattice Enumeration"

[Jaq+20] Jaques et al., "Implementing Grover Oracles for Quantum Key Search on AES and LowMC"

The results highlight vulnerabilities in certain post-quantum cryptographic schemes, demonstrating how and at what cost adversaries may exploit the structure of the schemes. At the same time, the findings indicate that attacks on hash-based signature schemes such as SPHINCS$^+$ as well as lattice-enumeration based attacks on key encapsulation mechanism are far from practical. Such attacks are limited by the high computational resources required, making attacks even with quantum computers infeasible. Understanding implications from these challenges allows to further advance the field of cryptography, resulting in more robust constructions.

Part III

QUANTUM-SECURE PROTOCOLS

Summary

Part II demonstrates how providing concrete resources to an adversary affects the security of post-quantum cryptography. Our results, for example the IND-CCA attack of Chapter 5, show that even if intractability assumptions remain computationally hard, some protocol designs have significant security flaws. At the same time, it appears that secure post-quantum cryptography can be achieved. An example are signatures on the basis of cryptographic hash functions (cf. Chapter 6), or key encapsulation mechanisms based on lattice assumptions[11] (cf. Chapter 7). While the results from Part II suggest that certain cryptographic algorithms may have lower security margins than expected, strong security properties can be achieved when instantiated correctly. As such, we now move on to incorporate post-quantum secure schemes in higher level protocols, possibly exchanging the existing conventionally secure cryptography. We begin by assuming that post-quantum secure algorithms exists, specifically signature schemes and key encapsulation mechanisms, and then investigate the impact of deploying these in a higher level protocol.

[11] And, of course, other assumptions that are outside the focus of this manuscript.

The adoption of quantum secure schemes comes with formidable challenges, as this necessitates the overhaul of existing protocols and the assessment of the security provided by quantum-secure intractability assumptions. For example, a transformation from classically to post-quantum security often entails to exchange the existing Discrete Logarithm (DLOG)-based Diffie–Hellman key-exchange with a post-quantum secure key encapsulation mechanism, particularly, one of the KEM candidates of the NIST post-quantum competition [Nat17]. However, the two components, Diffie–Hellman (DH) key exchange and KEM, can not be readily switched in common key agreement protocols without invalidating the security proofs — potentially resulting in loosing security properties. It is natural to ask...

[Nat17] National Institute for Standards and Technology, *Post-Quantum Cryptography Call for Proposals*

> *...what security properties remain intact when exchanging a Diffie–Hellman key exchange for a KEM?*

We answer this question for the L-band Digital Aeronautic Communication System (LDACS) protocol in Chapter 9.

In addition to reviewing the security of post-quantum cryptography Part II also strengthens the understanding of the cost of implementing quantum algorithms. In particular, Chapters 6 and 7 show that implementing quantum algorithms to attack cryptographic schemes can require an enormous amount of resources. This becomes even more apparent when imposing real-world constraints, such as quantum error correction, or a limit on the time that a quantum circuit can remain coherent. Unsurprisingly, attacking conventionally secure cryptography, such as factorization or discrete-logarithm based systems face a similar quantum computational overhead. While research suggests [GE21] that quantum computers capable of performing such attacks on conventional cryptography may be in reach in the coming decades, it also

[GE21] Gidney and Ekerå, "How to factor 2048 bit RSA integers in 8 hours using 20 million noisy qubits"

appears that such computation would be very expensive. For instance, *commercially* available quantum computers that can factor numbers such as 21 or 35 are in a price range of millions[12] of Euros. While such costs are expected to come down, if quantum computers are ever powerful enough for practical computation and manufactured for a larger market, it seems reasonable, that in the coming decades quantum computing remains costly. This raises the question of whether there are other ways to protect ourselves against quantum threats without having to replace all cryptographic primitives.

[12] Frankly, I have not found any *quotable* sources for this.

> *What security against quantum adversaries can be achieved without post-quantum cryptography?*

With the premise that quantum computation is possible, but remains expensive, surprisingly, not all protocols require a replacement of the underlying encryption schemes. For example, PAKE protocols can achieve the quantum-annoying property. That means, that the quantum resources that an adversary has to invest to break security of a protocol scales with number of password guesses, i. e., with a parameter that can be increased for more security in these schemes.

In support of addressing these questions, we first review computational security models for key agreement protocols in Chapter 8.

Contribution 4. To investigate the impact of deploying post-quantum cryptography in higher level protocols we analyze the LDACS protocol, which is designed to secure the civil aviation communication over the coming decades. The protocol features the possibility to secure the communication with either classically-secure, or post-quantum public-key cryptography. While the protocol is currently under standardization by the International Civil Aviation Organization [Int21], the modifications to the deployed protocol to enable the use of KEMs have not been analyzed formally, thus providing no guarantee that the desired security properties, such as entity and key authentication, hold with these new premises.

[Int21] International Civil Aviation Organization (ICAO), *ICAO - ANNEX 10 VOL III AMD 91 Aeronautical Telecommunications Volume III - Communications Systems (Part I - Digital Data Communication Systems; Part II - Voice Communication Systems)*

In Chapter 9, we investigate which security notions are preserved when the protocol is instantiated with (quantum-secure) key encapsulation mechanisms. We provide precise, explicit predicate based definitions of various security notions along with a detailed proof of each property. As such, we can show that the key-exchange of the LDACS protocol preserves its most important security properties, particularly entity authentication, providing the communication partner with the assurance that they talk to their intended peer, and key authentication, promising that (only) the two intended peers know the exchanged key. We prove this both in a computational "pen-and-paper" proof, as well as an automated symbolic-based proof system, which can be reproduced. The chapter is based on [TMM24b].

[TMM24b] Tiepelt, Martin, and Mäurer, "Post-Quantum Ready Key Agreement for Aviation"

Contribution 5. The phenomena of quantum-annoyingness was first observed during the CFRG PAKE standardization process [Cry19] in 2019, where it was found [Tho19] for one of the DDH-based candidates, that even if an attacker could solve discrete logarithms, they could not immediately recover the password. Instead, an attacker seemed to have to compute a DLOG for each guess of the password even during an offline dictionary attack:

[Cry19] Crypto Forum Research Group, *CFRG PAKE Standardization Process*

[Tho19] Thomas, *"Re: [Cfrg] Proposed PAKE Selection Process"*

this property was named “quantum-annoying”. If solving such a problem remains reasonably expensive, then a moderate level of security can still be achieved. While this does not give full quantum resistance, it allows to scale the cost of a quantum adversary with the entropy of the used password, potentially making quantum attacks substantially more expensive.

In Chapter 10, we show how to leverage this property in the setting of asymmetric PAKEs. The content of this chapter is based on [TES23b], where we proposed an augmentation to an existing PAKE protocol. While the original protocol is susceptible to a quantum attacker solving a single discrete logarithm, after our augmentation, we prove that an adversary has to solve a discrete logarithm for every password guess — thus achieving quantum-annoying-ness — while providing fall-back to the security of the DDH assumption.

[TES23b] Tiepelt, Eaton, and Stebila, “Making an Asymmetric PAKE Quantum-Annoying by Hiding Group Elements”

8

Security Models for Authenticated Key Exchange

This chapter reviews security models for authenticated key exchange that allow to prove desirable properties formally. This first Section 8.1 describes the model and Section 8.2 defines the properties used in Chapter 9 to prove the security for the aviation protocol. The last Sec. 8.3 and 8.4 recalls the security model of password authenticated key exchange in support of Chapter 10.

Parts of this chapter have been taken verbatim from our publications, i. e., [TES23b; TES23a; TMM24b; TMM24a].

8.1 Computational Security Model

We consider game-based security for authenticated key exchange in the Bellare-Rogaway Model [Brz+11] as formalized by [SFW19], the description of which features precise notation using predicates with explicit security properties. A short version of [SFW19] was published as [SFW20]. However, we refer to the full version as we believe this to be more accessible. We closely follow [SFW19, Sec. 2] to describe the model and its parameters.

[Brz+11] Brzuska et al., "Composability of bellare-rogaway key exchange protocols"

[SFW19] Saint Guilhem, Fischlin, and Warinschi, *Authentication in Key-Exchange: Definitions, Relations and Composition*

[SFW20] Saint Guilhem, Fischlin, and Warinschi, "Authentication in Key-Exchange: Definitions, Relations and Composition"

Setup: Identities and Protocol Executions. The security model is defined over a set $\mathcal{I} \subset \mathbb{N}$ of parties. Each party has a unique party identifier $i, j \in \mathcal{I}$. The total number of parties, $n = |\mathcal{I}|$, is fixed a priori. The parties are either an initiator I, or a responder R, i. e., $I, R \in \mathcal{I}$.

The set of authenticating parties is $\mathcal{S}$. We consider the case of mutual authentication for all parties, consequently we have $\mathcal{S} = \mathcal{I}$. We work in the pre-specified peer model, which means, that each party is aware of their own identifier, and also knows their *intended* partner's unique identifier, which is also called the pid.

These parties engage in a protocol $\Pi = (\textit{key-generation}, \Pi)$, defined by an initial *key-generation* as well an algorithm Π corresponding to the locally executed procedure of each party. The execution of *key-generation* is not part of the proof framework and not considered further. Each individual execution of a protocol by an entity $i \in \mathcal{I}$ is called a local session, associated with a local session $\ell = (i, j, k)$ where k denotes that this is the k-th session between party i and party j, where j is the intended communication partner. We refer to i as ℓ.id and to j as ℓ.pid. The total number l of local sessions is bound a priori, such that $k \leq l$.

GAME STATES AND BOOKKEEPING. The security of a protocol Π is defined via an experiment called a *game*, where the adversary engages in protocol executions and adaptively interacts with or corrupts parties. [SFW19] introduce five lists to track the individual states of protocol executions and the game.

[SFW19] Saint Guilhem, Fischlin, and Warinschi, *Authentication in Key-Exchange: Definitions, Relations and Composition*

1. The list SST of protocol-related session states stores for each local session $\ell = (i, j, k)$ a signature key pair (vk_i^{sig}, sk_i^{sig}) of party i as well as their intended partner's verification key vk_j^{sig}. The list further stores a set of variables:
   - accept $\in \{\text{true}, \text{false}, \perp\}$ in initialized to $\perp$ and is set when the session terminates, meaning the party does not receive or send any further messages.
   - The session identifier sid and the session-key K.
   - The variables kcid, ecid $\in \{0,1\}^* \cup \perp$, which are initialized to $\perp$, and set as soon as the local session accepts. The variables are set during a session and then remain invariant for the remainder of the session. The kcid's purpose is to unambiguously determine the value of the session key before the session accepts while the ecid determines the value of the session identifier.
   - The model features a key confirmation flag *kconf* $\in \{\text{none}, \text{almost}, \text{full}\}$, denoting which form of confirmation should be achieved.
2. The list LST of game-related local session states corresponds to how the adversary interacts with the sessions throughout the game. For each session, it stores whether the owner and peer of a session are honest or corrupted, $\delta_{\text{ownr}}, \delta_{\text{peer}} \in \{\text{honest}, \text{corrupt}\}$, and similarly the state of a session $\delta_{\text{sess}} \in \{\text{fresh}, \text{revealed}\}$. Parties and sessions are honest and fresh by default, and may change their state depending on the adversary's interaction.
3. The list corresponding to the game execution state EST contains information, $\{(i, vk_i^{sig}, sk_i^{sig}, \delta_i)\}_i$, denoting which parties have been corrupted (independently of sessions).
4. LSID is the list of valid local session identifiers.
5. The list MST holds information related to specific security notions.

8.1.1 *Security Goal: Authentication*

EXPERIMENT AND ADVERSARIAL INTERACTION. Security is defined via an experiment where an adversary interacts with a game. In the setting of quantum-secure communication the adversary A is a quantum polynomial-time algorithm which interacts through classical queries with the game. The set of all queries is $Q = \{\text{SEND}, \text{REVEAL}, \text{CORRUPT}\}$. On input $\text{SEND}(\ell, m)$, the game processes Π on m on behalf of local session ℓ, and returns the result to the adversary. On input $\text{REVEAL}(\ell)$, the game sets $\ell.\delta_{\text{sess}}$ to revealed, and returns ℓ.K to the adversary. On input $\text{CORRUPT}(i)$, the entity i is marked

as corrupt in variable δ_i. Then, for any session owned by that entity $(i, \cdot, \cdot)$, and any session where it is listed as a peer $(\cdot, i, \cdot)$, the entity is marked as corrupt, i. e., the game sets $\delta_{\text{ownr}} = \text{corrupt}$ (respectively for δ_{peer}). Finally, the secret signing key sk_i^{sig} is returned to the adversary.

The game returns responses to the adversary according to an algorithm χ, which evaluates a query $q \in Q$ and the game state. [SFW19] further allow the adversary to submit invalid queries, determined by a predicate *Valid*, for which χ is not executed. We note that such queries are not relevant in our setting. The state of each game G comprises the five lists from the previous paragraph.

[SFW19] Saint Guilhem, Fischlin, and Warinschi, *Authentication in Key-Exchange: Definitions, Relations and Composition*

The experiment then consists of three phases: In a first phase of the experiment, key pairs for all entities are generated and variables initialized as previously defined. In the second phase, the adversary may interact with the sessions and parties by submitting queries q to the game, which are processed by χ. In the last phase, after the adversary send all their queries and terminated, the game evaluates a predicate on the state $b \leftarrow P(\text{SST}, \text{LST}, \text{EST}, \text{LSID}, \text{MST})$ and outputs the bit $b \in \{0, 1\}$. The adversary wins the game, if $b = 0$.

QUANTIFICATION. With this setup, a security notion is fully defined by a predicate P. A protocol achieves a security property if for all polynomial-time adversaries A the predicate P evaluates to 1 except with negligible probability, i. e.,

$$Adv^{G}_{A,\Pi,\mathcal{I},\mathcal{S}} = \mathbb{P}\left[\text{Exp}^{G}_{A,\Pi,\mathcal{I},\mathcal{S}}(1^{\lambda}) = 0\right] \in negl\,,$$

where $\text{Exp}^{G}_{A,\Pi,\mathcal{I},\mathcal{S}}$ corresponds to running the game G with protocol Π, parties $\mathcal{I}$ and authenticating parties $\mathcal{S}$ against the adversary A. We note that for the remaining paper we only write "adversary" instead of "polynomial-time adversary" for simplification, and security holds against quantum polynomial-time adversaries if the respective components are post-quantum secure.

8.1.2 *Security Goal: Secrecy*

EXPERIMENT AND ADVERSARIAL INTERACTION. To define a notion for secrecy via the predicate BRSEC, [SFW19, Sec. 5] provide a modified game G_{BRSEC}, where the adversary's goal is to distinguish a real and a random session key. The modified game state includes EST, SST, LST. The state MST includes two additional bits and a challenge session. The bit b_{test} determines whether the adversary is provided with a real key from a session, or whether they are provided with a random key. The bit b_{guess} will hold the adversary's guess. The challenge session ℓ_{test} will correspond to the adversary's choice of which session to test. To set these states, the game provides two additional queries to the adversary. On input $\text{TEST}(\ell)$, the game sets the test session $\ell_{\text{test}} \leftarrow \ell$ and returns $\text{K} = \ell.\text{K}$ if $b_{\text{test}} = 1$, or a random key K from the key space if $b_{\text{test}} = 0$. On input $\text{GUESS}(b)$, the bit $b_{\text{guess}} \leftarrow b$ is updated. Additionally, the game restricts the adversary to submit only a single TEST query. The BRSEC predicate evaluates to the bit b_{guess} as in Definition 8.2.12.

QUANTIFICATION. Let $G_{\text{BRSEC}}^{b_{\text{test}}=b}$ denote the BR-secrecy game during which b_{test} has the value b. Due to the distinguishing nature of the BR-secrecy game, the adversarial advantage is defined as

$$Adv_{A,\Pi,\mathcal{I},\mathcal{I}}^{G_{\text{BRSEC}}}(1^\lambda) := \left| \mathbb{P}\left[\text{Exp}_{A,\Pi,\mathcal{I},\mathcal{I}}^{G_{\text{BRSEC}}^{b_{\text{test}}=0}}(1^\lambda) = 1\right] - \mathbb{P}\left[\text{Exp}_{A,\Pi,\mathcal{I},\mathcal{I}}^{G_{\text{BRSEC}}^{b_{\text{test}}=1}}(1^\lambda) = 1\right] \right| .$$

8.2 PREDICATES FOR AUTHENTICATED KEY EXCHANGE

We recall the definitions of the various security properties and the predicates introduced by [SFW19] with the simplifications relevant to this manuscript. Note that all predicates are quantified over the list of valid local session identifiers, i. e., $\forall \ell \in \text{LSID}$. All predicates are to be used in conjunction with the game as defined in Section 8.1.1 unless otherwise stated. The predicates SAMEKEY, SAMEKCID, SAMEECID and PARTNERED are defined to enable the readability of the subsequent predicates:

[SFW19] Saint Guilhem, Fischlin, and Warinschi, *Authentication in Key-Exchange: Definitions, Relations and Composition*

Definition 8.2.1 (SAMEKEY, abridged from [SFW19, Def. 2.3]). *SAMEKEY*$(\ell, \ell') \Leftrightarrow \ell.K = \ell'.K \neq \perp$ *for distinct sessions* ℓ, ℓ'.
The definitions for SAMEKCID and SAMEECID are analogous.

Definition 8.2.2 (PARTNERED, abridged from [SFW19, Def. 2.1]). *PARTNERED*$(\ell, \ell') \Leftrightarrow \ell \neq \ell'$ *and* $\ell.\text{sid} = \ell'.\text{sid} \neq \perp$.

The MATCH property promises, intuitively, that two parties engaging with the same session identifier also derive the same key K and kcid, and that the latter guarantees the computation of identical keys. Additionally, the property promises that each local session is partnered with at most one other local session.

Definition 8.2.3 (Match predicate, abridged from [SFW19, Def. 2.3]).

$$\begin{aligned} \textit{MATCH} \Leftrightarrow \forall \ell, \ell', \ell'' : \ & (\textit{PARTNERED}(\ell, \ell') \wedge \ell.K \neq \perp \neq \ell'.K) \Longrightarrow \textit{SAMEKEY}(\ell, \ell') \\ & \wedge (\textit{PARTNERED}(\ell, \ell') \wedge \ \ell.\text{kcid} \neq \perp \neq \ell'.\text{kcid}) \Longrightarrow \textit{SAMEKCID}(\ell, \ell') \\ & \wedge (\textit{PARTNERED}(\ell, \ell') \wedge \ \textit{PARTNERED}(\ell, \ell'')) \Longrightarrow \ell' = \ell'' \\ & \wedge (\textit{SAMEKCID}(\ell, \ell') \wedge \ell.K \neq \perp \neq \ell'.K) \Longrightarrow \textit{SAMEKEY}(\ell, \ell') \end{aligned}$$

ENTITY AND KEY AUTHENTICATION. Implicit entity authentication holds, if any party is guaranteed that for each of its sessions, only its intended partner has a matching sid.

Definition 8.2.4 (Implicit Entity Authentication, abridged from [SFW19, Def. 3.1]).

$$\textit{iENTAUTH} \Leftrightarrow \forall \ell \textit{ that accept} : (\forall \ell' : \textit{PARTNERED}(\ell, \ell') : \ell'.id = \ell.pid)$$

Entity confirmation improves this by providing assurance that there exists another session with the same sid ("full"), or the same ecid ("almost full"), where for the latter matching ecids eventually result in the same sid if the respective session terminates successfully.

Definition 8.2.5 (Full Entity Confirmation, abridged from [SFW19, Sec. 9.1]).

$$\textit{fENTCONF} \Leftrightarrow \forall \ell \textit{ that accept} \wedge \ell.\delta_{\text{peer}} = \text{honest} : \exists \ell' : \textit{PARTNERED}(\ell, \ell')$$

Definition 8.2.6 (Almost-Full Entity Confirmation, abridged from [SFW19, Sec. 9.1])**.**

$$\textsc{afEntConf} \Leftrightarrow \forall\ \ell \textit{ that accept} \wedge \ell.\delta_{\text{peer}} = \text{honest} : \exists \ell' : \\ (\textsc{SameECID}(\ell, \ell') \wedge (\ell'.\text{sid} \neq \bot \implies \textsc{Partnered}(\ell, \ell')))$$

"Almost" full key confirmation promises, that parties engaging with a honest peer have assurance, that their peer derives the same key. We note that "full" key confirmation is implied by another, stronger predicates later on, and thus not explicitly defined.

Definition 8.2.7 (Almost-Full Key Confirmation, abridged from [SFW19, Def. 3.4])**.**

$$\textsc{afKeyConf} \Leftrightarrow \forall \ell \textit{ that accept} \wedge \ell.\delta_{\text{peer}} = \text{honest} : \\ \exists \ell' : \textsc{SameKCID}(\ell, \ell') \wedge ((\ell'.K \neq \bot) \Rightarrow \textsc{SameKey}(\ell, \ell'))$$

The following four primary security notions of [SFW19] are the main terms in our Theorem 3 of Chapter 9. (Almost) Full explicit entity authentication promises that for honest, authenticating parties only the intended partner has a matching sid and there exists at least one such session with matching sid (respectively ecid for "almost").

[SFW19] Saint Guilhem, Fischlin, and Warinschi, *Authentication in Key-Exchange: Definitions, Relations and Composition*

Definition 8.2.8 (Full Explicit Entity Authentication, abridged from [SFW19, Sec. 9.1])**.**

$$\textsc{fexEntAuth} \Leftrightarrow \forall \ell \textit{ that accept} : (\forall \ell' : \textsc{Partnered}(\ell, \ell') : \ell'.id = \ell.pid) \\ \wedge (\ell.\delta_{\text{peer}} = \text{honest} \implies \exists \ell' : \textsc{Partnered}(\ell, \ell'))$$

Definition 8.2.9 (Almost-Full Explicit Entity Authentication, abridged from [SFW19, Sec. 9.1])**.**

$$\textsc{afexEntAuth} \Leftrightarrow \forall \ell \textit{ that accept} : (\forall \ell' : \textsc{Partnered}(\ell, \ell') : \ell'.id = \ell.pid) \\ \wedge (\ell.\delta_{\text{peer}} = \text{honest} \implies \exists \ell' : \textsc{SameECID}(\ell, \ell') \\ \wedge (\ell'.\text{sid} \neq \bot \implies \textsc{Partnered}(\ell, \ell')))$$

Similarly, (almost) full explicit key authentication promises that only the intended partner has sessions with matching key and that at least one such session with matching key exists (respectively kcid).

Definition 8.2.10 (Full Explicit Key Authentication, abridged from [SFW19, Def. 3.5])**.**

$$\textsc{fexKeyAuth} \Leftrightarrow \forall \ell \textit{ that accept} : (\forall \ell' : \textsc{SameKey}(\ell, \ell') : \ell'.id = \ell.pid) \\ \wedge (\ell.\delta_{\text{peer}} = \text{honest} \implies \exists \ell' \textit{ such that } \textsc{SameKey}(\ell, \ell'))$$

Definition 8.2.11 (Almost-Full Explicit Key Authentication, abridged from [SFW19, Def. 3.6])**.**

$$\textsc{afexKeyAuth} \Leftrightarrow \forall \ell \textit{ that accept} : (\forall \ell' : \textsc{SameKey}(\ell, \ell') : \ell'.id = \ell.pid) \\ \wedge (\ell.\delta_{\text{peer}} = \text{honest} \implies \exists \ell' : \textsc{SameKCID}(\ell, \ell') \\ \wedge (\ell'.K \neq \bot \implies \textsc{SameKey}(\ell, \ell')))$$

Finally, we define the predicate to achieve secrecy as used in the game in Section 8.1.2:

Definition 8.2.12 (BR-Secrecy, abridged from [SFW19, Def. 5.1, 5.2])**.** *The BRSec predicate is defined as follows:*

$$\begin{aligned}
&\textbf{\textit{if}} && \mathrm{MST}.\ell_{\mathrm{test}} \neq \perp \wedge \ell.\delta_{\mathrm{ownr}} = \ell.\delta_{\mathrm{peer}} = \mathrm{honest} \wedge \ell.\delta_{\mathrm{sess}} = \mathrm{fresh} \\
&&& \wedge \forall \ell' \in \mathrm{LST} : \textsc{Partnered}(\ell, \ell') \Rightarrow \ell'.\delta_{\mathrm{sess}} = \mathrm{fresh} \\
&\textbf{\textit{then}} && \mathit{BRSec} \leftarrow \mathrm{MST}.b_{\mathit{guess}} \\
&\textbf{\textit{else}} && \mathit{BRSec} \leftarrow 0
\end{aligned}$$

Remark 13. *In [SFW19, Def. 5.2], the BRSec predicate has value $\perp$ in some cases. Since the adversarial advantage does not distinguish whether BRSec has the value 0 or $\perp$, we omit this detail.*

[SFW19] Saint Guilhem, Fischlin, and Warinschi, *Authentication in Key-Exchange: Definitions, Relations and Composition*

8.3 SECURITY OF PASSWORD AUTHENTICATED KEY EXCHANGE

We consider the model of Bellare, Pointcheval and Rogaway (BPR00) [BPR00] for security of an asymmetric password-authenticated key exchange protocol, where the notation is unified with the model of [Brz+11] from Section 8.1. The setting is similar to that of authenticated key exchange (cf. Section 8.1) with modification to the identities, game states, queries and bookkeeping:

[BPR00] Bellare, Pointcheval, and Rogaway, "Authenticated Key Exchange Secure against Dictionary Attacks"

[Brz+11] Brzuska et al., "Composability of bellare-rogaway key exchange protocols"

There are two distinct sets of parties, the clients C, and the servers S, Each such party takes as inputs a unique identifier and a long-term secret. The client's long-term secret is a *low entropy* password π; the server's long-term secrets are the credentials $\mathit{cred}_S[C]$, that are established during a *Registration* phase which is not considered in detail. More formally:

1. The list SST of protocol-related session states stores for each local session $\ell = (i, j, k)$ a password π of party C, or the credentials $\mathit{cred}_S[C]$ for a party S. Note that the identifier j corresponds to the partner identifier in the original BPR00 model [BPR00]. Further, it stores a set of variables similar to the setting in Section 8.1: A variable accept, session identifier sid and session-key K $\in \{0,1\}^* \cup \perp$, the latter of which are initialized to $\perp$, and set as soon as the local session accepts. The model does not include flags denoting any form of confirmation that should be achieved.

2. The lists LST, LSID and MST are identical to the setting in Section 8.1.

3. The list corresponding to the game execution state EST contains information, $\{(i, \pi, \mathit{cred}_S[C], \delta_i)\}_i$, denoting which parties have been corrupted (independently of sessions).

QUANTUM-ANNOYING Additionally to BPR00 model, we consider the notion of quantum-annoying'ness in the generic group model which we call *QA-BPR*: A protocol is said to be quantum annoying, if a classical adversary who has the additional ability to solve discrete logarithms can break the protocol only by solving a distinct discrete logarithm for each guess of the password. While not fully resistant to attacks by quantum computers, a quantum-annoying protocol could offer some resistance to quantum adversaries for whom discrete logarithms are *relatively* expensive.

In this setting, the adversary gets access to the group operation $\circ$ and a discrete logarithm oracle DLOG. In Section 8.4 we describe how the generic group model can be utilized to quantify queries to the discrete logarithm oracle.

Remark 14. *We wish to emphasize for the reader that the "quantum annoying" security notion is an intermediate notion below fully quantum-resistant. One limitation of the quantum annoying security notion is that it has a narrow view of quantum capabilities: by using a formalism in the generic group model with a discrete logarithm oracle, we are effectively assuming that the only quantum operation an adversary will do is run Shor's algorithm, which is certainly less than the full power available to a polynomial time quantum computer.*

Even just considering security against quantum computers running Shor's algorithm, a protocol "secure" in the quantum-annoying model is still vulnerable to attacks by quantum computers, it is just that the attack scales in the size of the password space. This leads to the question of the cost of computing a discrete logarithm on a quantum computer. While it is impossible to predict the efficiency of quantum computers in the far future, current research suggests that the first generations of quantum computers capable of solving cryptographically relevant discrete logarithm problems will require significant resources in order to do so [Roe+17; GM19; GE21; PV23]. These estimates are undoubtedly coarse and may be off by several orders of magnitude, but it is plausible that even for early cryptographically relevant quantum computers, computing a single discrete logarithm will not be cheap, and that computing millions of discrete logarithms to find the password in a quantum-annoying PAKE may be prohibitively expensive.

[Roe+17] Roetteler et al., "Quantum Resource Estimates for Computing Elliptic Curve Discrete Logarithms"

[GM19] Gheorghiu and Mosca, *Benchmarking the quantum cryptanalysis of symmetric, public-key and hash-based cryptographic schemes*

[GE21] Gidney and Ekerå, "How to factor 2048 bit RSA integers in 8 hours using 20 million noisy qubits"

[PV23] Parker and Vermeer, *Estimating the Energy Requirements to Operate a Cryptanalytically Relevant Quantum Computer*

EXPERIMENT AND ADVERSARIAL INTERACTION. The experiment is the same as for secrecy (cf. Section 8.1.2), except that the set of queries $Q = \{\text{EXECUTE}, \text{SEND}, \text{REVEAL}, \text{CORRUPT}\}$ now includes EXECUTE, which mimics a passive observation of a session by the adversary. This is necessary, because in the setting of password-authenticated key exchange the adversarial advantage is quantified over the number of SEND queries. CORRUPT queries return the parties long-term secrets.[1]

[1] In the setting of PAKE's this is known as *weak corruption*.

QUANTIFICATION. The security is defined by the adversary's probability to decide if they received a session key or a random string after submitting a TEST query to a *fresh* instance.

In the setting of quantum-annoying-ness, *fresh* means that neither the session nor any *partnered* session may be corrupted. That means, on input $\text{TEST}(\ell)$, the game checks if the predicate from Definition 8.3.2 is fulfilled, and if not, sets the bit $b_{\text{guess}} = 0$. However, first the definition of partnered has to be adjusted to include some of the conditions formerly covered in the MATCH notion for authenticated key exchange (cf. Definition 8.2.3):

Definition 8.3.1 ($\text{PARTNERED}^{\text{PAKE}}$)**.**

$$\begin{aligned}\textit{Partnered}^{\textit{PAKE}}(\ell, \ell') \Leftrightarrow \textit{Partnered}(\ell, \ell') \ \wedge \ \textit{SameKey}(\ell, \ell') \\ \wedge\ \ell.\textit{accept} = \ell'.\textit{accept} = \textit{true} \\ \wedge\ \ell.\textit{pid} = \ell'.\textit{id} \wedge \ell'.\textit{pid} = \ell.\textit{id}\end{aligned}$$

Definition 8.3.2 (QA-fresh)**.**

$$QA-\text{fresh}(\ell) \Leftrightarrow \ell.\delta_{\text{ownr}} = \text{honest} \\ \wedge \forall \ell' : \textsc{Partnered}^{PAKE}(\ell, \ell') \Rightarrow \ell'.\delta_{\text{ownr}} = \text{honest}$$

Let $G^{b_{\text{test}}=b}_{\text{QA-BPR}}$ denote the quantum-annoying game during which b_{test} has the value b. Let q_{Send} be the number of online interactions and q_{Dlog} the number of discrete logarithm queries. In the BPR00 model the adversary's advantage is defined as

$$Adv^{G_{\text{QA-BPR}}}_{A,\Pi,\mathcal{I},\mathcal{I}}(1^\lambda) := \left| \mathbb{P}\left[\text{Exp}^{G^{b_{\text{test}}=0}_{\text{QA-BPR}}}_{A,\Pi,\mathcal{I},\mathcal{I}}(1^\lambda) = 1\right] - \mathbb{P}\left[\text{Exp}^{G^{b_{\text{test}}=1}_{\text{QA-BPR}}}_{A,\Pi,\mathcal{I},\mathcal{I}}(1^\lambda) = 1\right] \right| .$$

A protocol is called quantum annoying, if

$$Adv^{G_{\text{QA-BPR}}}_{A,\Pi,\mathcal{I},\mathcal{I}}(1^\lambda) \leq \frac{q_{\text{Send}} + q_{\text{Dlog}}}{N} + \epsilon \,,$$

with a password space of size N and ϵ negligible in the security parameter.

8.4 QUANTUM ANNOYING-NESS IN THE GENERIC GROUP MODEL.

In the normal generic group model there is a multiplicative public representation of group elements taken uniformly from $\{0,1\}^\lambda$, and an additive secret representation in $\mathbb{Z}_p$. The public representations have no intrinsic structure, and so any information about the group is obtained through the group operation oracle. Let $\langle g \rangle = \mathbb{G}$ be a generic group of size p with group operation $\circ$. When $g_w \circ g_v$ is queried, for example $(g^v, g^w) \mapsto g^{v+w}$, a table T_{ggm} is used to retrieve the secret representations of g_v and g_w, $v, w \in \mathbb{Z}_p$. Then $v + w \pmod p$ is the secret representation of g^{v+w}. If g^{v+w} has already been given a public representation that is returned. Otherwise, a uniformly random string is sampled from $\{0,1\}^\lambda$, assigned as a new public representation to g^{v+w} in the table T_{ggm}, and provided back to the querier. Table 8.1 gives an example of queries defining public and secret representations of the generic group model.

TABLE 8.1: Examples for simulation of queries to the generic group model group operation and the ideal cipher. The queries are in order from top to bottom. The public representations returned from the oracle are uniformly random strings that contain no information beyond what was given in the query and particularly are sampled independently.

Oracle	Label	Public Repr.	Secret Repr.
Initialization	g_0	11 ... 101	1
$\circ(g_0, g_0)$	$g_0 g_0$	01 ... 001	2
IC	g_a	11 ... 001	χ_a (unif. random index a)
IC	g_b	10 ... 111	χ_b (unif. random index b)
$\circ(g_a, g_b)$	$g_a g_b$	01 ... 000	$\chi_a + \chi_b$

time ↓

Similarly, the discrete logarithm oracle $\textsc{Dlog} : \mathbb{G} \times \mathbb{G} \to \mathbb{Z}_p$ takes as input two group elements and outputs the discrete logarithm. The query

$\textsc{Dlog}(g_v, g_w)$ can be responded to by looking up g_v and g_w in T_{ggm} and returning $w \cdot v^{-1} \pmod p$.

The generic group model is a powerful tool, but limited in its ability to reason about whether the adversary's interactions with the discrete logarithm oracle are sufficient to determine $\textsc{Dlog}(g, g_t)$ for a specific group element g_t. Naturally, if they have made exactly this query, the discrete logarithm is known. But other queries, such as $\textsc{Dlog}(g, g_t^2)$, are also sufficient to make $\textsc{Dlog}(g, g_t)$ knowable.

The framework of [ES21] simulates the generic group model in such a way that such specific statements can be made. Let $G_1, G_2, \ldots G_\mu$ be a collection of (public representations of) group elements whose discrete logarithm (with respect to the group generator g) are of potential interest to the adversary. When we maintain the group, rather than imbuing these group elements with specific secret representations in $\mathbb{Z}_p$, we instead denote each as a formal independent variable $\chi_1, \ldots, \chi_\mu$. Group operations now correspond to addition over a *vector space* of dimension $\mu + 1$. For example, in computing $(G_1 \circ G_1) \circ G_2$ we would calculate the secret representation as $2\chi_1 + \chi_2$ and give this a unique public representation in $\{0,1\}^\lambda$. Thus, secret representations can now be written as a linear combination of the χ_i variables, i. e., $\alpha_0 + \sum_i \alpha_i \chi_i$.

Table 8.2 gives an example for simulating $\textsc{Dlog}$ queries, showing the cases where the $\textsc{Dlog}$ can take a uniformly random value in $\mathbb{Z}_p$ and when the $\textsc{Dlog}$ between two elements is already defined (but has not yet been queried).

[ES21] Eaton and Stebila, "The "Quantum Annoying" Property of Password-Authenticated Key Exchange Protocols"

TABLE 8.2: Example simulation of the $\textsc{Dlog}$ oracle with public and secret representations as in Table 8.1 and with a generic group of size $p = 29$. The vector $\vec{s}$ is sampled at random such that $D\vec{s} = \vec{r}$, and δ is the returned discrete logarithm as described in the $\textsc{Dlog}$ oracle. For the vector $\vec{s}$ only the relevant entries are displayed as integers in $\mathbb{Z}_{29}$, and all other entries are marked with $*$, denoting a random integer which does not impact the computation of δ. The first $\textsc{Dlog}$ query adds a zero-vector to D, since the $\textsc{Dlog}$ (i. e., 2) was already defined by the secret representation. Since D and $\vec{r}$ are empty, there is no restriction on the choice of $\vec{s}$. The 2nd and 3rd query return a random integer δ in the solution space. The response to the 4th query was already constrained by the 2nd and 3rd query, which can also be seen by checking that the vector corresponding to $\vec{b}$ was already in the row span of D before this query was received.

					Add to D			Add to $\vec{r}$
$\textsc{Dlog}(g_v, g_w)$	g_v	g_w	$\vec{s}$	δ	χ_1	χ_a	χ_b	
$\textsc{Dlog}(g, g_1)$	1	2	$(*,*,*)$	2 (forced by relations)	0	0	0	0
$\textsc{Dlog}(g, g_a)$	1	χ_a	$(*,13,*)$	13 (random in $\mathbb{Z}_p$)	0	-1	0	-13
$\textsc{Dlog}(g_a, g_c)$	χ_a	$\chi_a + \chi_b$	$(*,13,4)$	$\frac{(13+4)}{13} \equiv 8 \mod 29$ (random in $\mathbb{Z}_p$)	0	7	-1	0
$\textsc{Dlog}(g, g_b)$	1	χ_b	$(*,13,4)$	4 (forced by relations)	0	0	-1	-4

time ↓

Thinking about how these secret representations interact with the $\textsc{Dlog}$ oracle is how we can start to reason about what discrete logarithms are. Say the adversary queries $\textsc{Dlog}(A, B)$, and the secret representation of A is $\alpha_0 + \sum \alpha_i \chi_i$ (respectively with β for B). If the adversary is given the response

δ (so that $A^\delta = B$), this imposes a constraint on our variables. Specifically, it says that $\delta\left(\alpha_0 + \sum \alpha_i\chi_i\right) = \beta_0 + \sum \beta_i\chi_i$, which we can rewrite as

$$\sum_{i=1}^{\mu}(\delta\alpha_i - \beta_i)\chi_i = \beta_0 - \delta\alpha_0. \tag{8.1}$$

This linear constraint lets us define an equivalence relation: if two secret representations are the same 'modulo' the linear constraints imposed by responses to DLOG, they should have the same public representation. Consequently, if, modulo these constraints, a secret representation χ_i is equivalent to some $a \in \mathbb{Z}_p$, then $\text{DLOG}(g, G_i)$ has taken on a definite value a, whether or not it was actually queried. Otherwise, it can still take on any possible value.

By taking the coefficients of the χ_i variables in Equation 8.1 we can construct a matrix D and a vector $\vec{r}$ (we write vectors as column vectors), so that the set of constraints is easily summarized as $D\vec{\chi} = \vec{r}$. Similarly, a secret representation $a_0 + \sum a_i\chi_i$ can be written as the pair $(a_0, \vec{a})$. In more detail, the equivalence relation can be defined as follows:

Definition 8.4.1 ($(D, \vec{r})$-equivalent)**.** *For group elements g_a, g_b with secret representation $(a_0, \vec{a})$ and $(b_0, \vec{b})$, we say that g_a is $(D, \vec{r})$-*equivalent to *g_b if there exists an $\vec{\omega} \in \mathbb{Z}_p^{q_D}$ such that $\vec{\omega}^T D = \vec{a}^T - \vec{b}^T$ and $\vec{\omega}^T\vec{r} = b_0 - a_0$.*

Note that this is indeed an equivalence relation (reflexivity is proven by taking $\vec{\omega} = \vec{0}$, symmetry is proven by taking $-\vec{\omega}$, and transitivity is proven by taking $\vec{\omega}_1 + \vec{\omega}_2$). The reason that this definition gives us what we want is that when it is satisfied, we have that $b_0 - a_0 = \vec{\omega}^T\vec{r} = \vec{\omega}^T D\vec{\chi} = (\vec{a}^T - \vec{b}^T)\vec{\chi} = \vec{a}^T\vec{\chi} - \vec{b}^T\vec{\chi}$, telling us that $a_0 + \sum a_i\chi_i = b_0 + \sum b_i\chi_i$, as we expect. We can now describe how the $\mathbb{G}$ and the DLOG oracle are simulated in full detail. Note that the simulation is not *efficient* [ES21, Sec. 4], since the simulation requires to search through all previous queries to check if a linear relationship exists. However, the purpose of the framework is to give an information-theoretic bound (in the generic group model) relative to the number of discrete logarithm queries to define a specific discrete logarithm, thus the exact efficiency is not relevant.

[ES21] Eaton and Stebila, "The "Quantum Annoying" Property of Password-Authenticated Key Exchange Protocols"

SIMULATION OF $\text{DLOG}(g_V, g_W)$. If g_V or g_W do not exist in T_{ggm}, then abort. Otherwise, let $(v_0, \vec{v}), (w_0, \vec{w})$ be secret representations of g_V, g_W respectively. Sample a random vector $\vec{s}$ such that $D\vec{s} = \vec{r}$ and compute $\delta = (w_0 + \langle\vec{w}, \vec{s}\rangle)/(v_0 + \langle\vec{v}, \vec{s}\rangle) \mod p$. Add the row $\delta\vec{v}^T - \vec{w}^T$ to D, and value $w_0 - \delta v_0$ to vector $\vec{r}$. Then δ is the discrete logarithm that is returned. This corresponds to [ES21, Alg. 2].

SIMULATION OF $\circ(g_V, g_W)$. If g_V or g_W do not exist in T_{ggm}, then abort. Otherwise, for the secret representations $(v_0, \vec{v})$, $(w_0, \vec{w})$, let $(z_0, \vec{z}) = (v_0 + w_0, \vec{v} + \vec{w})$. If z appears in T_{ggm}, return the corresponding public representation. Otherwise, check if there exists an entry $(f_0, \vec{f})$ of T_{ggm} that is $(D, \vec{r})$-equivalent to $(z_0, \vec{z})$. If so, return the public representation of that entry. If no such $(D, \vec{r})$-equivalent entry exists, sample a new public representation, add the entry $T_{\text{ggm}}[g_Z] = (z_0, \vec{z})$ and return g_Z. This corresponds to [ES21, Alg. 5].

With these two simulations, we can prove Lemma 3, which is a generalization of [ES21, Lemma 1] and an instantiation of which is used in a game hop in Section 10.2.2.

[ES21] Eaton and Stebila, "The "Quantum Annoying" Property of Password-Authenticated Key Exchange Protocols"

Lemma 3 (Unique Solutions)**.** *Let g_a and g_b be public representations of group elements, with corresponding secret representations $(a_0, \vec{a}), (b_0, \vec{b})$. Let $(D, \vec{r})$ be the current set of constraints on discrete logarithms. Then the discrete logarithm of g_b with respect to g_a is defined if and only if $[\vec{b}^T | b_0]$ is in the rowspace of the matrix* $\left[\begin{array}{c|c} -D & \vec{r} \\ \hline \vec{a}^T & a_0 \end{array}\right]$.

Proof. The discrete logarithm is defined if and only if there exists an α such that g_a^α is $(D, \vec{r})$-equivalent to g_b. By definition, this is the same as the existence of $\alpha, \vec{\omega}$ such that $\vec{\omega}^T D = \alpha \vec{a}^T - \vec{b}^T$, and $\vec{\omega}^T \vec{r} = b_0 - \alpha a_0$. We can rewrite this relation as

$$\left[\vec{b}^T \mid b_0\right] = \left[-\vec{\omega}^T D + \alpha \vec{a}^T \mid \vec{\omega}^T \vec{r} + \alpha a_0\right] = \left[\frac{\vec{\omega}}{\alpha}\right]^T \left[\begin{array}{c|c} -D & \vec{r} \\ \hline \vec{a}^T & a_0 \end{array}\right]. \quad (8.2)$$

This establishes that if the discrete logarithm is defined, $[\vec{b} \mid b_0]$ is indeed in the rowspace, and if it is in the rowspace that the discrete logarithm is defined (and equal to the α value that is the scalar for the 'a' row). □

Corollary 9. *Let g_b be the public representation of a group element and $(b_0, \vec{b})$ the corresponding secret representation. Let g be the generator of the group, which has secret representation $(1, \vec{0})$. Then the discrete logarithm of g_b with respect to g is defined if and only if $\vec{b}$ is in the row span of D.*

Proof. We apply Lemma 3 with $\vec{a} = \vec{0}$. Since the zero vector cannot affect the row span, we can conclude that $\vec{b}^T$ must be in the row span of D.

For the other direction we know that there exists some $\vec{\omega}$ such that $\vec{\omega}^T D = \vec{b}^T$. Then we claim that the discrete logarithm between g and g_b is $b_0 + \vec{\omega}^T \vec{r}$. This is because we want $(b_0 + \vec{\omega}^T \vec{r}, \vec{0})$ to be $(D, \vec{r})$-equivalent to g_b, and indeed we can see that $-\vec{\omega}$ satisfies $-\vec{\omega}^T D = -\vec{b}^T$ and $-\vec{\omega}^T \vec{r} = b_0 - (b_0 + \vec{\omega}^T \vec{r})$ as desired. □

9

Post-Quantum-Ready Authenticated Key Agreement for Aviation

Parts of this chapter are verbatim from our publications [TMM24b; TMM24a].

Current Air Traffic Management suffers from a lack of digitalization, for example analogue voice communications, and insufficient cybersecurity measures in aeronautical datalinks and applications. In Europe, these challenges are investigated within the Single European Sky Air traffic management Research joint undertaking [Sin23]. As a result, a new long-range terrestrial Air-Ground communication protocol was proposed, which must include measures to "[Provide] a secure channel [...] to ensure authentication and integrity of [...] message exchange" [Aer21; Int21]. Currently, the L-band Digital Aeronautical Communications System (LDACS) [SES23] is under standardization by the International Civil Aviation Organization [Int21], which provides a framework for air-ground communication between civilian aircraft and ground stations. A placement of LDACS in such communication networks is depicted in Figure 9.1.

At the heart of the LDACS security architecture lies a mutually authenticated key exchange, instantiating either an ISO key agreement protocol [ISO21] with DH, or a similar protocol where the DH component is exchanged for a key encapsulation mechanism. This is motivated by the understanding, that the public key exchange schemes undergoing standardization as part of the NIST post-quantum competition [Nat22] are all KEMs.

After successful key agreement, the resulting key material will be used to secure LDACS control-plane and user-plane data. Examples of user-plane data include applications such as the Controller-Pilot Data Link Communications, Automatic Dependent Surveillance-Contract or Ground-Based Augmentation System. The first is used for communications of clearances, route-changes, speed or radio frequency assignments and other mission critical data. The second application allows the automatic surveillance of the airspace and informs air traffic controllers about the position, levels and headings of aircraft. The third enables fully automatic landings, without the need of a human intervention [SES23]. Since the landing is naturally a dangerous phase of any flight the lack of a human in the loop as a control instance requires a substantial amount of trust in these data.

Authenticated Key Agreement. Authenticated key agreement enables two or more parties to secretly agree on a key, additionally assuring the parties that they are interacting with their intended peer. Common frameworks

[Sin23] Single European Sky ATM Research Joint Undertaking, *Single European Sky ATM Research*

[Aer21] Aeronautical Radio, Incorporated (ARINC), *Internet Protocol Suite (IPS) for Aeronautical Safety Services Part 1 Airborne IPS System Technical Requirements*

[Int21] International Civil Aviation Organization (ICAO), *ICAO - ANNEX 10 VOL III AMD 91 Aeronautical Telecommunications Volume III - Communications Systems (Part I - Digital Data Communication Systems; Part II - Voice Communication Systems)*

[SES23] SESAR JU, *LDACS A/G Specification, Edition 01.01.00, Template Edition 02.00.05, Edition date 25.04.2023*

[ISO21] ISO copyright office, *ISO/IEC 11779-3*

[Nat22] National Institute for Standards and Technology, *NIST: Selected Algorithms 2022*

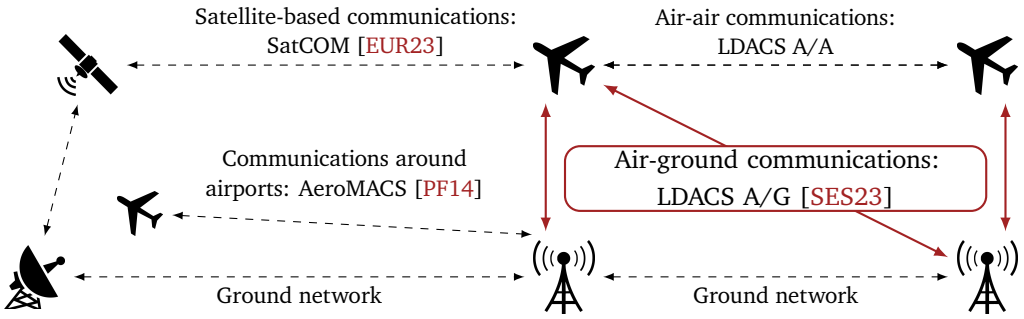


FIGURE 9.1: Placement of LDACS (solid, burgundy lines) in commercial aviation communication networks. AeroMACS is a short-range system to operate around airports. SatCOM allows aircraft to receive navigational support data.

to prove the security of key agreement protocols are BR-[Brz+11] or SK-security [CK02], with adversaries that have the capabilities to send or modify any messages during protocol execution. Additionally, adversaries may either learn the initial state of a party (weak corruption), or even learn intermediate states during execution of the protocol (strong corruption). We review common protocols and their security that relate to this manuscript. A broader overview can be found in [BMS20].

[Brz+11] Brzuska et al., "Composability of bellare-rogaway key exchange protocols"

[CK02] Canetti and Krawczyk, "Security Analysis of IKE's Signature-Based Key-Exchange Protocol"

[BMS20] Boyd, Mathuria, and Stebila, *Protocols for Authentication and Key Establishment, Second Edition*

The Station to Station protocol [WVW92] combines a DH key exchange with a public-key infrastructure signature scheme. At first, the initiator sends a public DH value to a responder, who generates their own DH values, computes the secret key, and responds with their own DH value and a signature on both values. The responder encrypts their return message with the shared key. In the last pass, the initiator sends their own signature on the DH public values encrypted under the shared key. A variant of the Station to Station protocol removes the encryption of the signatures and either adds a Message Authentication Code (MAC) on the public DH values, or includes the identity of the intended partner into the signature. This construction allows to achieve security against weak corruption. The ISO/ IEC 11770-3 protocols, specifically ISO KAM-7 [ISO21, Sec. 11.7] pick up at this latter construction, but add an identifier for the intended partner to the signature and an additional message authentication code to the message flow, lifting the protocol to provide security against strong corruption.

[WVW92] Whitfield, Van Oorshot, and Wiener, "Authentication and authenticated key exchanges"

[ISO21] ISO copyright office, *ISO/IEC 11779-3*

Finally, the Internet Key Exchange Version 2 (IKEv2) [Kau+14a], which is implemented in IPsec and virtually all Internet-Protocol-based secure communications (e. g. Virtual Private Networks), follows the SIGn-and-MAc (SIGMA) approach [Kra03]. The SIGMA approach combines the public-key infrastructure-based signature, the message authentication code and the encryption of the two using the shared key, and was proven secure against strong corruption.

[Kau+14a] Kaufman et al., *Internet Key Exchange Protocol Version 2*

[Kra03] Krawczyk, "SIGMA: The 'SIGn-and-MAc' Approach to Authenticated Diffie-Hellman and Its Use in the IKE-Protocols"

POST-QUANTUM CRYPTOGRAPHY. The progress in quantum technology makes a potential threat of large scale universal quantum computers to cryptography undeniable. To enable secure communication in the future, quantum secure alternatives to Diffie–Hellman-like key exchanges that do not rely on the hardness of the discrete logarithm problem, have been proposed. The most prominent of such alternatives are the post-quantum schemes that are currently undergoing standardization as part of the NIST post-quantum

competition [Nat22]. These schemes are based on KEMs, meaning that they cannot readily be combined with existing protocols that build on a Diffie–Hellman-like key exchange structure, thus invalidating security guarantees. On the other side, any protocol that achieves security based on a *generic* notion for key encapsulation mechanisms could be instantiated with any of the NIST post-quantum schemes, providing a quantum-secure protocol.

[Nat22] National Institute for Standards and Technology, *NIST: Selected Algorithms 2022*

KEY AGREEMENT IN THE WAKE OF QUANTUM COMPUTERS. The previously mentioned key agreement protocols have been, originally, proven secure under the Decisional Diffie–Hellman assumption [CK02; Cre11], thus are based on only classically secure components. A first formal analysis of IKEv2 assuming a post-quantum secure variant of DH was conducted by [Gaz+21], in which case the original proof would carry over, except under the post-quantum DH assumption. An alternative approach was taken by [Flu+20], who show that IKEv2 is "post-quantum secure" if pre-shared keys are used to negotiate new key material. Further, [Bin+19] provide a detailed analysis of hybrid combiners using Diffie–Hellman values and KEMs, along with a fine-grained analysis of adversarial power. They provide a multi-stage security proof for an independent authenticated key exchange based on key encapsulation mechanism from a SIGMA approach. Further, [Pei14] sketched the proof of a variant of SIGMA, which exchanged the Diffie–Hellman key exchange for a key encapsulation mechanism.

[CK02] Canetti and Krawczyk, "Security Analysis of IKE's Signature-Based Key-Exchange Protocol"

[Cre11] Cremers, "Key Exchange in IPsec Revisited: Formal Analysis of IKEv1 and IKEv2"

[Gaz+21] Gazdag et al., "A Formal Analysis of IKEv2's Post-Quantum Extension"

[Flu+20] Fluhrer et al., *Mixing Preshared Keys in the Internet Key Exchange Protocol Version 2 (IKEv2) for Post-quantum Security*

[Bin+19] Bindel et al., "Hybrid Key Encapsulation Mechanisms and Authenticated Key Exchange"

[Pei14] Peikert, "Lattice Cryptography for the Internet"

CONTRIBUTION & OBJECTIVE. In this work we bridge the gap between classically and quantum secure authenticated key agreement when using any (post-quantum) IND-CPA secure key encapsulation mechanism. Our main contribution is the analysis of the LDACS key agreement protocol [SES23, Sec. C.8.2], which supports to deploys a key encapsulation mechanism instead of a Diffie–Hellman key exchange. To the best of our knowledge previous *investigations* of the LDACS MAKE protocol (for example [Mäu+21; MGS21; MG22]) rely on heuristic arguments and do not provide security proofs or a formal analysis of the security guarantees in the computational and symbolic model.

[SES23] SESAR JU, *LDACS A/G Specification, Edition 01.01.00, Template Edition 02.00.05, Edition date 25.04.2023*

[Mäu+21] Mäurer et al., "A Secure Cell-Attachment Procedure of LDACS"

[MGS21] Mäurer, Gräupl, and Schmitt, *Cybersecurity for the L-band Digital Aeronautical Communications System (LDACS)*

[MG22] Mäurer and Grundner-Culemann, *Formal Verification of the LDACS MAKE Protocol*

We provide a simplified and idealized variant of the real-world protocol in Section 9.1.1, which we then prove to be secure in a computational, game-based model [Brz+11; SFW19], by proving Theorem 3 in Section 9.2. Additionally, we provide an automated Tamarin proof in the symbolic model, the results of which are outlined in Section 9.3 assuming a Dolev-Yao attacker.

[Brz+11] Brzuska et al., "Composability of bellare-rogaway key exchange protocols"

[SFW19] Saint Guilhem, Fischlin, and Warinschi, *Authentication in Key-Exchange: Definitions, Relations and Composition*

Theorem 3. *Let kem be an IND-CPA secure key encapsulation mechanism except with advantage ε_{kem} and correctness $1-\delta_{kem}$, let sig be an EUF-CMA secure signature scheme except with advantage ε_{sig}, and PRF be an ϵ_{PRF} secure pseudo-random function. Let n be a bound on the number of parties, l a bound on the total number of sessions and λ a security parameter. Then the LDACS MAKE protocol (cf. Figure 9.3) provides (almost) full explicit entity authentication and (almost) full explicit key authentication, which implicitly includes BR-secrecy. Any quantum polynomial time bounded adversary has advantage of falsifying the notions of at most*

$$4n\varepsilon_{sig}+\frac{4l^2}{2^{2\lambda}}+5l\delta_{kem}+2l^2\cdot\left(\frac{4(\varepsilon_{kem}+\varepsilon_{PRF})}{\frac{1}{l^2}-2(\varepsilon_{kem}+\varepsilon_{PRF})-n\varepsilon_{sig}}+2(\varepsilon_{kem}+\varepsilon_{PRF})\right)+\frac{2}{2^{\lambda}}\,.$$

We provide a complete proof of Theorem 3, with modularized security notions and explicit reductions, giving the precise security loss. The protocol achieves security against polynomially bounded quantum adversaries if the respective instantiations are quantum secure, i. e., if the KEM and the signatures are instantiated with post-quantum secure schemes, and the PRF is secure with appropriate key length. Independently, we provide a model of the LDACS protocol in Tamarin, achieving security when assuming the cryptographic components to be *perfect*. The results of the latter confirm our findings for the computational proof of the LDACS protocol.

TECHNICAL OUTLINE. We consider the setting of two parties each with a unique id, an initiator "ground station" *GS* and a responder "air station" *AS*, engaging in a key agreement scheme. Every party has access to a public-key infrastructure, i. e., a long-term private signing key for an EUF-CMA secure signature scheme *sig* as well as a respective verification key to check signatures of an intended communication partner. Further, the parties have agreed on a cipher suite determining an IND-CPA secure key encapsulation mechanism *kem*, an EUF-CMA secure message authentication code MAC, and a pseudo-random function PRF.

The party *GS* initiates the protocol by generating a *fresh* KEM key pair and sending the public key pk_{kem} to *AS*. *AS* runs an encapsulation algorithm on pk_{kem} resulting in a key k and a ciphertext. A MAC key and a session key are derived from k using the pseudo-random function PRF. Then the *AS* generates a MAC tag and a signature of the ciphertext, the unique id of *GS* and pk_{kem} using the MAC key and their long-term signature key. Finally, the *AS* sends the ciphertext, the signature and the tag to *GS*. *GS* verifies the signature, decapsulates the ciphertext, derives the MAC key and a session key and verifies the MAC tag. If the MAC and signature verifications succeed, *GS* constructs their own signature and MAC tag of the ciphertext, the id of *AS* and the KEM public key, send both to *AS* and accepts the session key. *AS* accepts their session key if the tag and the signature verify correctly. The above protocol resembles a simplification of the LDACS MAKE protocol as detailed in Section 9.1.1.

The security of the simplified key agreement protocol is captured in the Match-security model [BPR00]. We adopt the predicate-based framework of [SFW19] to model security goals and notions. Particularly, we prove all individual notions required to assemble Theorem 3 in Section 9.2, achieving the following properties:

[BPR00] Bellare, Pointcheval, and Rogaway, "Authenticated Key Exchange Secure against Dictionary Attacks"

[SFW19] Saint Guilhem, Fischlin, and Warinschi, *Authentication in Key-Exchange: Definitions, Relations and Composition*

Entity Authentication. The notion of (Almost) Full Explicit Entity Authentication captures the promise that each party is assured that they interact with their intended peer, and that their peer is aware of the identity of the party. Entity authentication holds due to the EUF-CMA security of the signature scheme *sig*.

Key Authentication. (Almost) Full Explicit Key Authentication gives the parties assurance that their intended peer and only their intended peer knows the secret key, which holds due to the EUF-CMA security of *sig*, the IND-CPA security of *kem* and the pseudo-randomness of PRF. This notion implies BR-secrecy with forward secrecy against weak corruption.

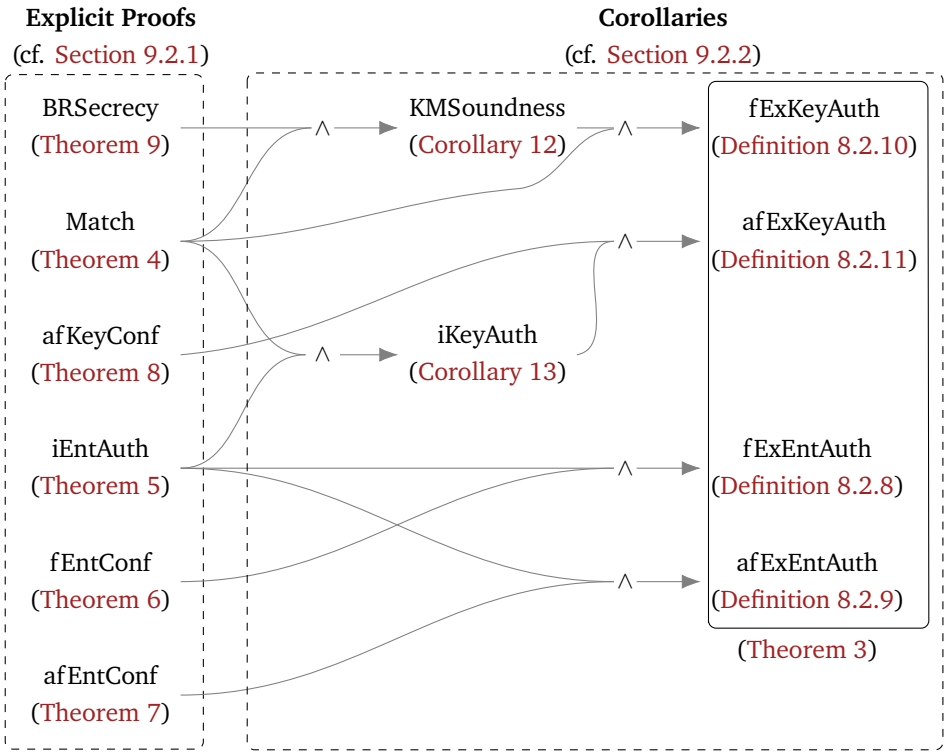


FIGURE 9.2: Overview of computational proof structure, showing how the explicit proofs of the security notions (as defined in Section 8.1) and the consequential security notions are related. The latter notions are implied by the logical conjunction of corresponding preceding notions. The solid box contains all security notions required for our main Theorem 3.

We provide a complete set of explicit proofs to various security notions of [SFW19] as outlined in Figure 9.2. The proof of BR-secrecy, which is implicitly used to assemble Theorem 3, partially follows the SK-security proof of the SIGMA protocol [Kra03; CK02], with the suggestions of [Pei14] for exchanging the DH values for a key encapsulation mechanism in the SIGMA protocol. The adaptions are marked accordingly.

Further, we provide a symbolic proof of mutual authentication, key confirmation and secrecy corresponding, the implementation and results of which are detailed in Section 9.3. The automated proof supports the findings of the computational proof.

[SFW19] Saint Guilhem, Fischlin, and Warinschi, *Authentication in Key-Exchange: Definitions, Relations and Composition*

[Kra03] Krawczyk, "SIGMA: The 'SIGn-and-MAc' Approach to Authenticated Diffie-Hellman and Its Use in the IKE-Protocols"

[CK02] Canetti and Krawczyk, "Security Analysis of IKE's Signature-Based Key-Exchange Protocol"

[Pei14] Peikert, "Lattice Cryptography for the Internet"

Remark 15. *The message authentication code used in the LDACS protocol has no impact on the security loss in Theorem 3. This is not surprising, since the KEM-based LDACS protocol features a signature similar to ISO-type protocols, which are known [Kra03, Sec. 4] to achieve security with DH-based instantiations by including the identities into the signatures. This is also in line with the main goal of LDACS to provide security against attackers that only corrupt the long-term keys.*

9.1 A SIMPLIFIED LDACS PROTOCOL

In this section, we provide a simplified protocol resembling the LDACS MAKE. The LDACS MAKE protocol [SES23, Sec. C.8.2] features two individual definitions that either use a Diffie–Hellman key exchange, or a KEM based key exchange. The first instantiates the standardized *Key Agreement Mechanism*

[SES23] SESAR JU, *LDACS A/G Specification, Edition 01.01.00, Template Edition 02.00.05, Edition date 25.04.2023*

7 (KAM-7) [ISO21, Sec. 11.7], and inherits all security properties. As such, we do not provide any further analysis of this instantiation. The latter substitutes the Diffie–Hellman key exchange for a construction using a key encapsulation mechanism, and thus does not instantiate ISO KAM-7, as this requires to be instantiated with a commutative function F: The session key is computed as $k = F(r_i, F(r_j, g)) = F(r_j F(r_i, g))$ [ISO21, Sec. 10.2], where r_i is a random value provided by one party, and r_j by the other party. The variant with the key encapsulation, however, computes the session key from $k, c \leftarrow \textsc{Encaps}(pk^i_{kem}, r_j)$ and $k \leftarrow \textsc{Decaps}(c)$, and thus does strictly not instantiate ISO KAM-7. As a consequence, the KEM-based protocol does not inherit the security properties of the ISO KAM-7. An examination of the security objectives [Int23; SES23], which are in understanding of the official security requirements for aeronautical communications set by the International Civil Aviation Organization (ICAO) [Aer21; Int21], is provided in this context. In Section 9.1.2 we compare our results to other authenticated key agreements.

[ISO21] ISO copyright office, *ISO/IEC 11779-3*

[Int23] International Civil Aviation organization, *CHAPTER 13 L-Band Digital Aeronautical Communications System (LDACS)*

[SES23] SESAR JU, *LDACS A/G Specification, Edition 01.01.00, Template Edition 02.00.05, Edition date 25.04.2023*

[Aer21] Aeronautical Radio, Incorporated (ARINC), *Internet Protocol Suite (IPS) for Aeronautical Safety Services Part 1 Airborne IPS System Technical Requirements*

[Int21] International Civil Aviation Organization (ICAO), *ICAO - ANNEX 10 VOL III AMD 91 Aeronautical Telecommunications Volume III - Communications Systems (Part I - Digital Data Communication Systems; Part II - Voice Communication Systems)*

9.1.1 *Key Agreement Protocol*

The LDACS MAKE protocol is split into two phases: In the *cell entry* stage, the two parties ground station *GS* and air station *AS* exchange miscellaneous information over an insecure channel with no attempt to achieve any security. At the end of the cell entry, both parties are aware of an intended partner associated by an identifier id_{GS} and id_{AS}, as well as a cipher suite used in the instantiation of the subsequent stage. Additionally, the *AS* can send a flag signaling that they do not have the ground station's signature public key vk^{sig}_{GS}, which can then be sent (in plain) by *GS*. We note that no security must hold for this initial phase, and this is not modeled in the protocol. Instead, we assume the intended entity identifiers id_{GS}, id_{AS} and respective signature keys from the public key infrastructure to be an input to the respective party. In the second stage, the two parties engage in a mutual authenticated key agreement protocol to exchange a secret that is used to derive multiple keys. This second stage is the 3-round MAKE protocol between two parties, the KEM-variant of which is the focus our analysis.

The protocol described in Figure 9.3 is a simplifications of the second stage from Section 9.1.1: Both parties take as input a unique identifier pid for an intended partner, their own unique identifier id, the public key vk^{sig}_{pid} of the intended partner, as well as the secret signing key sk^{sig}_{id}, the latter two of which are provided by the public key infrastructure. After completing the protocol, if the parties accept, then they have a session key K and assurance that only their intended partner has knowledge of this key. The desired security properties are discussed below, the achieved security notions along with the respective proof in Section 9.2. We prove the computational security of these notions in the game-based model [Brz+11; SFW20] for authenticated key exchange, and verify the security properties in the symbolic model using the Tamarin prover.

[Brz+11] Brzuska et al., "Composability of bellare-rogaway key exchange protocols"

[SFW20] Saint Guilhem, Fischlin, and Warinschi, "Authentication in Key-Exchange: Definitions, Relations and Composition"

Simplifications. The LDACS MAKE protocol specification [SES23, Sec. C.8.2] details the instantiation of the key derivation as well as identifier strings constructed from *constant* values that result in multiple individual

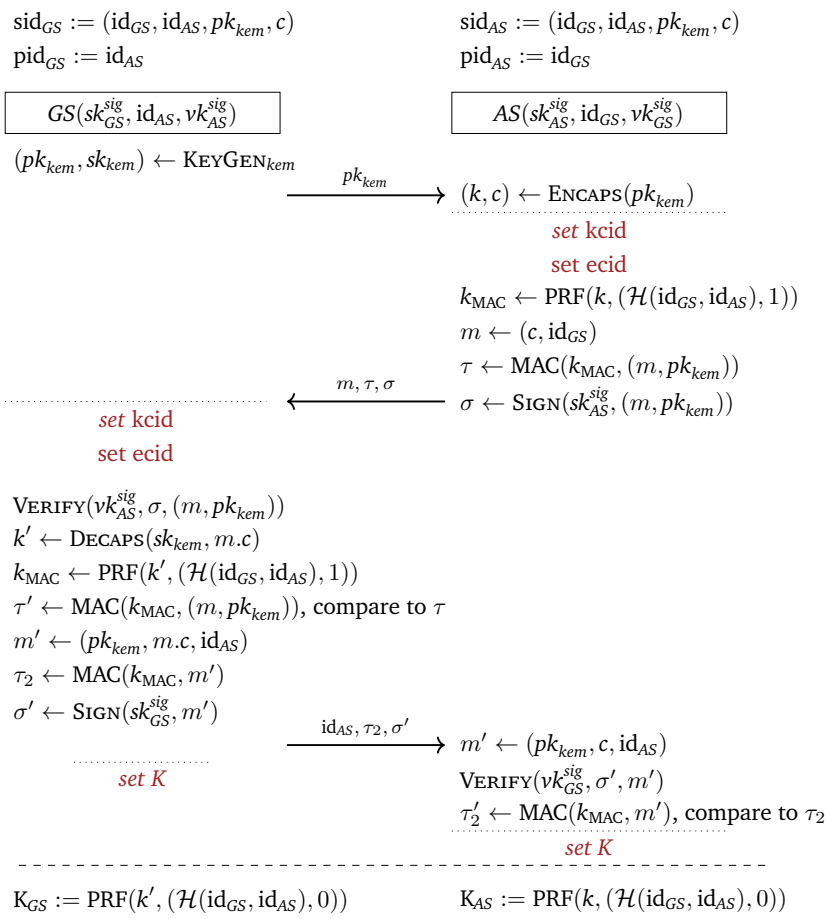


FIGURE 9.3: Simplified key agreement protocol between the ground station *GS* and the air station *AS*. We denote this protocol LDACS MAKE. If any of the verification VERIFY or comparison of MAC tags fail, the respective party aborts. The dotted lines with the burgundy identifiers are relevant for the computational proof in Section 9.2.

session keys, which are not relevant to security. In order to de-clutter the specification and provide a clear view of the protocol in question, we simplified the specification. In this paragraph we make these simplifications explicit and provide assumptions and justifications required for security of the protocol:

(1) The ground and air station are uniquely identified by $\text{id}_{GS} = (\text{UA}_{GS}, \text{SAC}_{GS})$ and $\text{id}_{AS} = (\text{UA}_{AS}, \text{SAC}_{AS})$ respectively.

(2) If $\text{id}_{GS}, \text{id}_{AS}$ are unique identifiers, then so is $\mathcal{H}(\text{id}_{GS}, \text{id}_{AS})$.

(3) Security achieved for the session key K_X also holds for multiple keys, i. e., "DCH session key", "Voice session key", "DCCH session key" and "Key encryption key".

(4) The composition of *HKDF* and $HMAC_{Hash}$ is a pseudo random function.

(5) Authentication Centers (AuC) are treated as if they were *GS*s.

Simplification (1) is purely cosmetic. Simplification (2) is justified because *SHA256*, $\mathcal{H}$, is believed to be a collision resistant hash function [Nat15]. Simplification (3) is justified because all keys are derived in the same manner,

[Nat15] National Institute for Standards and Technology, *Secure Hash Standard*

by computing $\mathrm{PRF}(k, \mathcal{H}(\mathrm{id}_{GS}, \mathrm{id}_{AS}))$ with constant strings as additional inputs. Simplification (4) is justified, because *HMAC* instantiated with *SHA256* is a pseudo-random function as shown by [Bel06]. This assumes that *HKDF* is instantiated accordingly. For Simplification (5), the specification mentions the possibility for *GS* to verify the *AS* certificates by communicating with an *Authentication Center* (AuC) over a secure channel [SES23, Sec. C.8.2] (see the "dashed" box). Since the communication with the AuC is considered secure we ignore this special case, and simply assume that the ground station can verify the correctness of the *AS* certificate locally. To the best of our knowledge our simplifications do not invalidate any security guarantees for the protocol as in the specification, since we only merge redundant components.

[Bel06] Bellare, "New Proofs for NMAC and HMAC: Security without collision-resistance"

[SES23] SESAR JU, *LDACS A/G Specification, Edition 01.01.00, Template Edition 02.00.05, Edition date 25.04.2023*

Security Goals. Aeronautical standards are split in two parts: First, the ICAO Standards And Recommended Practices (SARPs) define requirements of the technology; second, the specification described the technical realization of these requirements. As such, LDACS security goals are described in the SARPs [Int23], with the LDACS specification [SES23] detailing the protocols and other security measures.

[Int23] International Civil Aviation organization, *CHAPTER 13 L-Band Digital Aeronautical Communications System (LDACS)*

LDACS SARPs [Int23] set requirements for providing integrity protection [Int23, Sec. 13.8.2] and confidentiality [Int23, Sec. 13.8.4] of messages in transit, as well as mutual authentication [Int23, Sec. 13.8.5] between ground and air stations. These basic requirements were extended upon in the LDACS specification, i. e., that the key agreement fulfills mutual authentication and key establishment [SES23, Sec. C.8.2], as well as key confirmation [SES23, Sec. C.2, table 96] without specifying details. Further, the requirements ask for a "capability to prevent the propagation of intrusions within the LDACS access networks" [Int23, Sec. 13.8.8], which may be interpreted as forward secrecy, such that leakage of long-term keys does not invalidate security properties of other, honest parties.

9.1.2 *Comparison with other KEM-based Key Agreements*

SIGMA-style Authentication. The simplified protocol in Figure 9.3 is similar to the protocol suggested by [Pei14, Sec. 6.2], with adjustments to the messages used to generate the signature and the tag. [Pei14, Thm 6.1] states that their protocol is secure in the CK02 model but no complete proof is provided. Indeed, our proof of BR-secrecy partially follows the sketch they outlined, and we mark this in Section 9.2 where appropriate.

[Pei14] Peikert, "Lattice Cryptography for the Internet"

[Bin+19] present a compiler for hybrid authenticated key exchange, combining multiple KEMs and a SIGMA-style authentication scheme [Bin+19, Fig. 13]. Particularly, their scheme exchanges a KEM public key and ciphertext, which is subsequently authenticated along with a randomly sampled string (a nonce) along with the party identifiers. The authors show that their AKE construction provides BR-Match security and BR key secrecy in the BR93 model if the KEM is IND-CPA secure and if the signature scheme and the MAC are EUF-CMA secure, thus providing security against quantum adversaries, if the underlying components are quantum secure. In contrast, the protocol used in LDACS as well as our abstraction in Figure 9.3 uses ISO-style authentication and no nonces.

[Bin+19] Bindel et al., "Hybrid Key Encapsulation Mechanisms and Authenticated Key Exchange"

COMPARISON TO KEY EXCHANGE IN IKEV2. The minimal key exchange component [Kau+14b, Sec. 1.2] of the *Internet Key Exchange Version 2* [Kau+14a] consists of an initialization step IKE_SA_INIT and an authentication step IKE_AUTH, corresponding to the first three messages exchanged between the parties. Similarly to our setting, IKEv2 builds on an existing public key infrastructure and assumes that both parties can validate respective signatures. In the first step, the two parties perform a DH key agreement along with exchanging a randomly chosen nonce for each party. Using the shared secret and the two nonces they derive several keys [Kau+14a, Sec. 2.14] such as a session key and an additional key used for further authentication. In the second step, IKE_AUTH, each party sends an authenticated encryption with associated data (AEAD) of its own id and of a signature. The signature contains the message the party sent in the IKE_SA_INIT exchange, the nonce of the other party and the output of a PRF with the id of the party as input [Kau+14a, Sec. 2.15]. The AEAD and the PRF use some of the keys derived after IKE_SA_INIT.

[Kau+14b] Kaufman et al., *Minimal Internet Key Exchange Protocol Version 2*

[Kau+14a] Kaufman et al., *Internet Key Exchange Protocol Version 2*

The main differences between the key agreement of IKEv2 and the simplified protocol Figure 9.3 are that IKEv2 uses DH specifically and makes use of additional nonces. Further, they use an AEAD for the IKE_AUTH messages, and the signatures contain neither the id nor the DH exponential of the respective other party. They use a PRF instead of a MAC, the PRF output is contained in the signature and the PRF does not take the id of the other party as input.

Finally, IKEv2 [Kau+14a, Sec. 2.12] allows the reuse of DH exponents. Since the KEM's pk and c in the LDACS protocol also function as nonces (i. e., for each session they are chosen *freshly* uniformly at random and are unique except with negligible probability), it is essential that they are not reused, which is in line the LDACS specification.

9.2 COMPUTATIONAL PROOF

In this section we present the computational proof of security for the key agreement protocol described in Section 9.1. Specifically, we prove correctness of Implicit Entity Authentication (cf. Theorem 5) and BR secrecy (cf. Theorem 9) for both parties, Full entity confirmation (cf. Theorem 6) for the air station *AS*, and Almost Full Entity Confirmation (cf. Theorem 7) and Almost Full Key confirmation (cf. Theorem 8) for the ground station *GS*. From this, additional properties can be inferred, as detailed in Section 9.2.2.

9.2.1 *Explicit Proofs and Reductions*

We start by proving Lemma 4 which will be used throughout the subsequent proof. For all of the proof we set $\text{kcid} = \text{sid} = \text{ecid} = (\text{id}_{GS}, \text{id}_{AS}, pk_{kem}, c)$. For the remaining section we note that the key output space of the *kem* and the key input space of the PRF are identical. The same holds for the key space of the MAC and the co-domain of the PRF, i. e.,

$$\mathcal{K}_{kem} = \mathcal{K}_{\text{PRF}}$$
$$\mathcal{Y}_{\text{PRF}} = \mathcal{K}_{\text{MAC}}\,.$$

Further, the *kconf* flag as defined in Section 8.1 is set to "almost" for the groundstation *GS*,

Lemma 4. *Let A be an adversary interacting with the LDACS MAKE protocol. Let sig be an ε_{sig} secure signature scheme and n the number of parties participating in the protocol. Then the probability that an adversary can forge a signature of a party P that was not corrupted is bounded by*

$$\mathbb{P}\left[\text{SIG-FORGE}\right] \leq n \cdot \varepsilon_{sig} .$$

Proof. Let $\mathcal{B}_{sig}$ be an adversary that forges a signature in an execution of the LDACS MAKE protocol. We can construct a trivial reduction to an adversary that wins the EUF-CMA game: The adversary first guesses the party impersonated by $\mathcal{B}_{sig}$ and sets its verification key to the challenge key from the EUF-CMA game and queries the signing oracle from the EUF-CMA game to produce signatures for this party. If at any point $\mathcal{B}_{sig}$ forges a signature for the impersonated party, the reduction forwards this to the EUF-CMA game. This refers to a SIG-FORGE event. The probability that the reduction wins the EUF-CMA game is bounded by their ability to guess the impersonated party, i. e., $1/n$ and $\mathcal{B}_{sig}$ advantage to forge a signatures. Thus

$$\mathbb{P}\left[\text{SIG-FORGE}\right] \leq n \cdot \varepsilon_{sig} .$$

□

Theorem 4 (Match Security, cf. Definition 8.2.3)**.** *Let kem be δ_{kem} correct and l the number of local sessions. Any adversary's advantage to falsify the* MATCH *predicate in LDACS MAKE is bounded by*

$$\forall \textit{ adversaries } A : Adv^{G_{\text{MATCH}}}_{A,\Pi,\mathcal{I},\mathcal{I}}(1^\lambda) \leq \frac{l^2}{2^{2\lambda}} + l\delta_{kem} .$$

Proof. Recall that the kcid and session identifier are identical, kcid = sid, thus if ℓ, ℓ' are partnered and the kcid is set, then they also Match. Since ℓ.kcid $:= (\text{id}_{GS}, \text{id}_{AS}, pk_{kem}, c)$, they have the same view of the ciphertext c and the public key pk_{kem}, and thus the keys Match, except with probability δ_{kem}, the latter of which can occur on every session in question. Second, if the sessions ℓ', ℓ'' both share the same view of the sid with session ℓ, then $\ell' = \ell''$ except if there is a collision in both the public key and ciphertext, which happens with probability at most $1/2^{2\lambda}$ for each pair of instances. □

Theorem 5 (Implicit Entity Authentication, cf. Definition 8.2.4)**.** *Any adversary's advantage of falsifying the* IENTAUTH *predicate is zero:*

$$\forall \textit{ adversaries } A : Adv^{G_{\text{IENTAUTH}}}_{A,\Pi,\mathcal{I},\mathcal{I}}(1^\lambda) = 0$$

Proof. This is trivially fulfilled due to Matching sid $:= (\text{id}_{GS}, \text{id}_{AS}, pk_{kem}, c)$. □

Theorem 6 (Full Entity Confirmation, cf. Definition 8.2.5)**.** *Let sig be an ε_{sig} secure signature scheme. Any adversary's advantage of falsifying the* FENTCONF *predicate for the air station AS is bounded by*

$$\forall \textit{ adversaries } A : Adv^{G_{\text{FENTCONF}}}_{A,\Pi,\mathcal{I},\mathcal{I}}(1^\lambda) \leq n\varepsilon_{sig}$$

Proof. Let ℓ_{AS} be a session with peer ℓ_{GS}. Then ℓ_{AS} accepts after they received a signature $\sigma \leftarrow \textsc{Sign}(sk_{GS}^{sig}, (\mathrm{id}_{AS}, c, pk_{kem}))$ such that $\textsc{Verify}(vk_{GS}^{sig}, (\mathrm{id}_{AS}, c, pk_{kem}), \sigma)$ evaluates to 1. For a honest peer $\ell_{AS}.\mathrm{pid} = \ell_{GS}.\mathrm{id}$ it holds that $\ell_{GS}.\mathrm{sid} := (\ell_{GS}.\mathrm{id}, \ell_{AS}.\mathrm{id}, pk_{kem}, c)$, which is the same view of sid that ℓ_{AS} has, and thus they are partnered. For the successful verification we can construct a reduction to the EUF-CMA security of the signature analog to that of Lemma 4, such that the advantage is bounded by $n\varepsilon_{sig}$.

Remark: This does not hold for GS, since GS accepts after sending the last message, and AS potentially has not received the message yet, thus not set their sid *yet.* □

Theorem 7 (Almost-Full Entity Confirmation, cf. Definition 8.2.6)**.** *Let sig be an ε_{sig} secure signature scheme. The afEntConf predicate is fulfilled for the ground station GS if sig is a secure signature scheme:*

$$\forall \textit{ adversaries } A : Adv^{G_{\textsc{afEntConf}}}_{A,\Pi,\mathcal{I},\mathcal{I}}(1^\lambda) \leq n\varepsilon_{sig}$$

Proof. The proof is the same as for Theorem 6 with the argument over the ecid instead of the sid.

A *GS* accepts only, if they receive a signature $\sigma \leftarrow \textsc{Sign}(sk_{AS}^{sig}, (c, GS.\mathrm{sid}, pk_{GS}^{kem}))$ such that $\textsc{Verify}(vk_{AS}^{sig}, \sigma)$ evaluates to 1. This means, that if the signature was generated by an honest *AS*, then $AS.\mathrm{ecid} := (AS.\mathrm{id}, GS'.\mathrm{id}, c, pk^{kem,AS})$, which is the same as that of the *GS*. Since the ecid=sid, if the *AS*.sid is ever $\neq \perp$, then they have the same sid and are thus partnered. If the signature was not generated by an honest *AS*, we can construct a reduction to the EUF-CMA security of the signature analog to that of Lemma 4, such that the advantage is bounded by $\mathbb{P}[\textsc{Sig-Forge}] \leq n \cdot \varepsilon_{sig}$. □

Theorem 8 (Almost-Full Key Confirmation, cf. Definition 8.2.7)**.** *Let sig be an ε_{sig} secure signature scheme, and kem δ_{kem} correct. The afKeyConf predicate is fulfilled for the ground station GS, if sig is a secure signature scheme:*

$$\forall \textit{ adversaries } A : Adv^{G_{\textsc{afKeyConf}}}_{A,\Pi,\mathcal{I},\mathcal{I}}(1^\lambda) \leq n\varepsilon_{sig} + l\delta_{kem}$$

Proof. The proof is the same as for Theorem 6, but with kcid instead of sid. Additionally, afKeyConf requires ℓ, ℓ' to have the same key, which is the case if the kcids Match, except if the *kem* fails to decapsulate correctly. □

Theorem 9 (BR-Secrecy, cf. Definition 8.2.12)**.** *The BRSec predicate is fulfilled for both parties, if kem, PRF and sig are secure, i. e., $\varepsilon_{kem}, \varepsilon_{PRF}, \varepsilon_{sig} \leq negl(\lambda)$:*

$$\forall \textit{ adversaries } A : Adv^{G_{\textsc{BRSec}}}_{A,\Pi,\mathcal{I},\mathcal{I}}(1^\lambda) \leq \frac{4(\varepsilon_{kem} + \varepsilon_{PRF})}{\frac{1}{l^2} - 2(\varepsilon_{kem} + \varepsilon_{PRF}) - n\varepsilon_{sig}} + 2(\varepsilon_{kem} + \varepsilon_{PRF})$$

The proof of BRSec is closely related to that of SK-security from [CK02], where the DDH assumption has been exchanged for the IND-CPA security of a KEM as suggested by [Pei14]. The proof uses a set of supportive lemmas, Lemmas 5 to 8, most of which have a counterpart in the proof of [CK02]. The relation between the two proofs is summarized in Table 9.1.

In the following we provide the lemmas and sketch the proof of the Theorem 9. We highlight the parts where the DDH assumption has been

[CK02] Canetti and Krawczyk, "Security Analysis of IKE's Signature-Based Key-Exchange Protocol"

[Pei14] Peikert, "Lattice Cryptography for the Internet"

[TMM24a] Tiepelt, Martin, and Maeurer, *Post-Quantum Ready Key Agreement for Aviation*

TABLE 9.1: Relation between our proof of BR-secrecy and the proof of SK-security for the basic SIGMA protocol [CK02], where $DDH \rightsquigarrow kem$ denotes exchanging the DDH assumption for IND-CPA security of a KEM.

Our lemmas	SK-security proof [CK02]	Modifications / Comment
Lemma 4		Additional lemma
Lemma 5	Part of [CK02, Lem. 15]	More events, $DDH \rightsquigarrow kem$ (cf. [Pei16]).
Lemma 6	Part of [CK02, Lem. 15]	More events.
Lemma 7	Part of [CK02, Lem. 15]	More events.
Lemma 8	[CK02, Lem. 8 and 10]	More events, $DDH \rightsquigarrow kem$ (cf. [Pei16]).
Lemma 9	[CK02, Lem. 7]	Explicitly presume GUESS.
Lemma 10	[CK02, Lem. 11]	SAMEECID instead of Matching sessions.
Lemma 11	[CK02, Lem. 9]	Restated for LDACS MAKE.
Lemma 12	[CK02, Lem. 14]	Restated for LDACS MAKE.
Lemma 13	[CK02, Lem. 13]	Restated for LDACS MAKE.

exchanged for an assumption on the KEM security, and summarize the important parts. As the proof itself does not offer much insight, a complete version with proofs for all lemmas can be found in the full version [TMM24a].

Proof. The proof of BR-secrecy for the LDACS MAKE protocol shows that the probability of an adversary to distinguish the experiment G_{BRSEC} outputting a real or random key is negligible.

We follow the proof of SK-security for the basic SIGMA protocol [CK02], which defines five hybrids via five variants of a simulator $\hat{S}$. Each of the $\hat{S}$ variants guesses the test session and its partner in order to modify the generation of the session and mac keys for these two local sessions. If $\hat{S}$ guesses incorrectly or the adversary manipulates the communication between the two local sessions, then the simulator aborts. The first of the hybrids ("REAL") corresponds to the G_{BRSEC} game with real TEST output, unless the associated simulator $\hat{S}_{\text{REAL}}$ aborts. In the second, denoted "RPRF", the shared secret is exchanged for randomness which is used to compute the session and mac key. In the third game, "ALLR", both the session and the mac key are exchanged for randomness. The fourth game, "HYBR", reverses the change for the MAC key, to be generated from a random secret again. Finally, in the last game "RAND", the MAC key is reversed to be generated from the exchanged secret, and only the session key is set to a random value. Figure 9.4 shows how the five hybrids correspond to the individual lemmata to show indistinguishability of real and random outputs of the TEST query in G_{BRSEC}.

[CK02] Canetti and Krawczyk, "Security Analysis of IKE's Signature-Based Key-Exchange Protocol"

We define four different events $\mathcal{E} = \{\text{ABORT}, \text{AFFIRM}, \text{GUESS}, \text{SIG-FORGE}\}$ which occur during G_{BRSEC} or one of the hybrids:

- An ABORT event happens when the $\hat{S}$ simulator aborts, i. e., it guessed the test session or its partner incorrectly or the adversary manipulated the communication between the two local sessions.
- An AFFIRM event happens when the adversary sets b_{guess} to 1. Since the adversary acts as a distinguisher, we can express the indistinguishability of the hybrids via a negligible difference in the probabilities of the AFFIRM event.

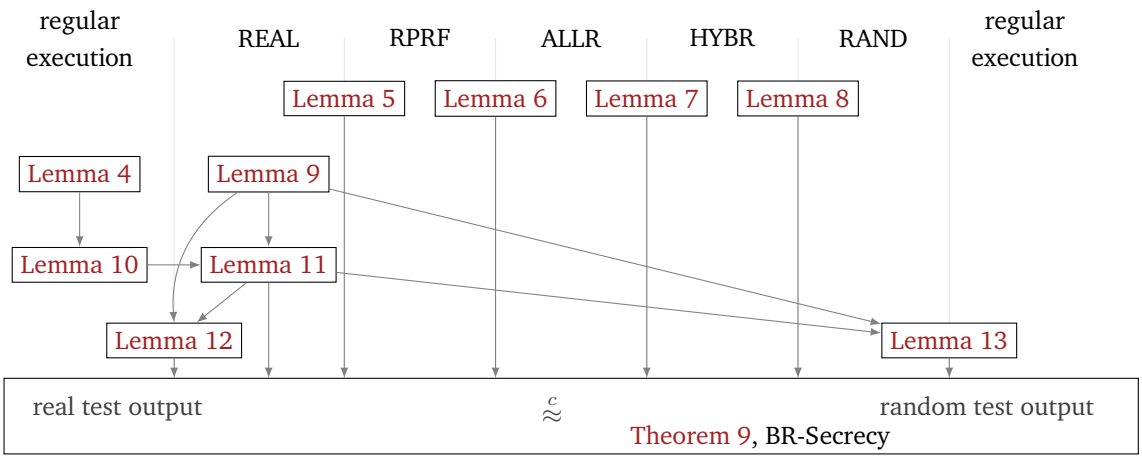


FIGURE 9.4: Overview of the lemmas in our proof of BR-Secrecy for the LDACS MAKE protocol. The five middle columns resemble the hybrid games according to the proof of SK-security in [CK02].

- A GUESS event happens when the $\hat{S}$ simulator guesses correctly.
- A SIG-FORGE event happens when the adversary successfully forges a signature from one of the protocol parties.

As a first step, our proof constructs Lemmas 5 to 8, which show that the probabilities of the four events only differ by a negligible amount between the different variants of the $\hat{S}$ simulator, where E corresponds to an event in the hybrid as in Figure 9.4. The full proof can be found in [TMM24a], and is omitted from this work, as they do not add any *interesting* insights.

[TMM24a] Tiepelt, Martin, and Maeurer, *Post-Quantum Ready Key Agreement for Aviation*

Lemma 5.

$$\forall \textit{ adversaries } A \; \forall E \subseteq \mathcal{E} : |\mathbb{P}[E_{REAL}] - \mathbb{P}[E_{RPRF}]| \leq \varepsilon_{kem}$$

Lemma 6.

$$\forall \textit{ adversaries } A \; \forall E \subseteq \mathcal{E} : |\mathbb{P}[E_{RPRF}] - \mathbb{P}[E_{ALLR}]| \leq \varepsilon_{PRF}$$

Lemma 7.

$$\forall \textit{ adversaries } A \; \forall E \subseteq \mathcal{E} : |\mathbb{P}[E_{ALLR}] - \mathbb{P}[E_{HYBR}]| \leq \varepsilon_{PRF}$$

Lemma 8.

$$\forall \textit{ adversaries } A \; \forall E \subseteq \mathcal{E} : |\mathbb{P}[E_{HYBR}] - \mathbb{P}[E_{RAND}]| \leq \varepsilon_{kem}$$

Since we get $\hat{S}_{\text{RPRF}}$ from $\hat{S}_{\text{REAL}}$ and $\hat{S}_{\text{HYBR}}$ from $\hat{S}_{\text{RAND}}$ by replacing the shared secret of the two guessed local sessions with a random value, and since in our case, the shared secret is a *kem* key, we can use the IND-CPA property of the *kem* to show the negligible difference between these simulators. We get $\hat{S}_{\text{ALLR}}$ from $\hat{S}_{\text{RPRF}}$ or from $\hat{S}_{\text{HYBR}}$ by replacing the outputs of the PRF with random values for the two guessed local sessions. Therefore, the negligible difference between these simulators follows from the PRF property.

Subsequently, we make a connection between the hybrid games and the G_{BRSEC} game. For this purpose, we first define Lemmas 9 to 11.

Lemma 9.

$$\forall \textit{ adversaries } A : \mathbb{P}\left[\textit{ABORT}_{\textit{REAL}} \wedge \textit{GUESS}_{\textit{REAL}}\right] = 0$$

Lemma 9 states that an ABORT and a GUESS event cannot coincide under an $\hat{S}_{\text{REAL}}$ simulator. Recall that a GUESS event implies that the simulator guesses the test session and its partner correctly. This especially means that a local session exists which will eventually become the partner of the test session, unless it aborts. In our case that local session has the same ecid as the test session. The ecid of a local session is composed of the identities of the two parties involved in the current protocol session as well as the public key and the ciphertext for the *kem*, which also are all the values and identities that the adversary could have manipulated via the exchanged messages. Thus, in case of a GUESS event, the simulator guesses the test session and its partner correctly and the adversary does not manipulate the communication between the two, i. e. the simulator will not abort.

Lemma 10.

$$\forall \textit{ adversaries } A\ \forall \ell \in \text{LSID}\ \exists \ell' \in \text{LSID} :$$
$$(\ell \textit{ accepts } \wedge\ \ell.\delta_{\text{peer}} = \text{honest} \Rightarrow \mathbb{P}\left[\neg \textit{SAMEECID}(\ell, \ell')\right] \leq n \cdot \varepsilon_{sig})$$

Lemma 10 shows that each local session which accepts and has an uncorrupted intended partner has the same ecid as some other local session. To ensure identical ecids, either of the two exchanged signatures is sufficient since it covers the public key and the ciphertext for the *kem* as well as the identity of the party which receives the signature. The identity of the signer is implicitly associated to the verification key.

The lemma is useful for Lemma 11, which establishes a noticeable lower bound for a GUESS event. The probability of correctly guessing two specific local sessions is trivially noticeable in the size of the set of all local sessions. However, the test session might not even have a partner, so we also need to use Lemma 10 to limit the probability for this case.

Lemma 11. *Similar to [CK02, Lem. 9]*

$$\forall \textit{ adversaries } A : \mathbb{P}\left[\textit{GUESS}_{\textit{REAL}}\right] \geq \frac{1}{l^2} - n\varepsilon_{sig}$$

Now we are able to show in Lemmas 12 and 13 that the "REAL" hybrid given a GUESS event is equivalent to G_{BRSEC} with real TEST output and that the "RAND" hybrid given a GUESS event is equivalent to G_{BRSEC} with random TEST output, except for a negligible probability. The reasoning is that, by Lemma 9, a GUESS event prevents the simulator from aborting. If the simulator does not abort, then all local sessions behave consistently and thus the adversary cannot distinguish between the simulation of the hybrids and G_{BRSEC}.

Lemma 12. *Similar to [CK02, Lem. 14]*

$$\forall \textit{ adversaries } A : \left|P_{\textit{REAL}}(A) - \mathbb{P}\left[\textit{AFFIRM}_{\textit{REAL}} | \textit{GUESS}_{\textit{REAL}}\right]\right| = 0$$

Lemma 13. *Similar to [CK02, Lem. 13] The following holds true:*

$$\forall \textit{ adversaries } A :$$
$$\left|P_{\textit{RAND}}(A) - \mathbb{P}\left[\textit{AFFIRM}_{\textit{RAND}} | \textit{GUESS}_{\textit{RAND}}\right]\right| \leq \frac{2(\varepsilon_{kem} + \varepsilon_{PRF})}{\frac{1}{l^2} - 2(\varepsilon_{kem} + \varepsilon_{PRF}) - n\varepsilon_{sig}}. \quad (9.1)$$

Finally, we combine all game hops from Lemmas 5 to 8, 12 and 13 to prove the BR-secrecy of the LDACS MAKE protocol. Although we now only consider the hybrids under the precondition that a GUESS event happens, the game hops of Lemmas 5 to 8 are still valid since Lemma 11 proved the probability of a GUESS event to be noticeable. □

9.2.2 *Consequential Security Properties*

Following the work of De Saint Guilhem et al. [SFW19], the Theorems 4 to 9 also imply the following security notions:

[SFW19] Saint Guilhem, Fischlin, and Warinschi, *Authentication in Key-Exchange: Definitions, Relations and Composition*

Corollary 10 (Full Explicit Entity Authentication, cf. Definition 8.2.8)**.** *The* FEXENTAUTH *predicate is fulfilled for the air station AS if sig is a secure signature scheme:*

$$\forall \textit{ adversaries } A : Adv^{G_{\text{FEXENTAUTH}}}_{A,\Pi,\mathcal{I},\mathcal{I}}(1^\lambda) \leq n \cdot \varepsilon_{sig}$$

FEXENTAUTH follows from IENTAUTH ∧ FENTCONF ⇔ FEXENTAUTH [SFW19, Prop. 9.3.] and from Theorems 5 and 6.

Corollary 11 (Almost-Full Explicit Entity Authentication, cf. Definition 8.2.9)**.** *The* AFEXENTAUTH *predicate is fulfilled for the ground station GS if sig is a secure signature scheme:*

$$\forall \textit{ adversaries } A : Adv^{G_{\text{AFEXENTAUTH}}}_{A,\Pi,\mathcal{I},\mathcal{I}}(1^\lambda) \leq n \cdot \varepsilon_{sig}$$

AFEXENTAUTH follows from IENTAUTH ∧ AFENTCONF ⇔ AFEXENTAUTH, which can be proven in an analogous way to [SFW19, Thm. 3.1.], and from Theorems 5 and 7.

Corollary 12 (Key-Match Soundness)**.** *The* KMSOUNDNESS *predicate is fulfilled for both parties if kem, PRF and sig are secure:*

$$\begin{aligned}&\forall \textit{ adversaries } A : Adv^{G_{\text{KMSOUNDNESS}}}_{A,\Pi,\mathcal{I},\mathcal{I}}(1^\lambda)\\ &\leq l^2 \cdot \left(\frac{4(\varepsilon_{kem}+\varepsilon_{PRF})}{\frac{1}{l^2}-2(\varepsilon_{kem}+\varepsilon_{PRF})-n\varepsilon_{sig}} + 2(\varepsilon_{kem}+\varepsilon_{PRF})\right) + \left(\frac{l^2}{2^{2\lambda}} + l\delta_{kem}\right) + \frac{1}{2^\lambda}\end{aligned}$$

KMSOUNDNESS follows from Theorems 4 and 9 and from [SFW19, Theorem 5.1.], which states that, with session key space $\mathcal{K}_\Pi$, the following holds:

$$\begin{aligned}&\forall \text{ adversaries } A \; \exists \text{ adversaries } \mathcal{B}_1, \mathcal{B}_2 :\\ &Adv^{G_{\text{KMSOUNDNESS}}}_{A,\Pi,\mathcal{I},\mathcal{I}}(1^\lambda) \leq l^2 \cdot Adv^{G_{\text{BRSEC}}}_{\mathcal{B}_2,\Pi,\mathcal{I},\mathcal{I}}(1^\lambda) + Adv^{G_{\text{MATCH}}}_{\mathcal{B}_1,\Pi,\mathcal{I},\mathcal{I}}(1^\lambda) + \frac{1}{|\mathcal{K}_\Pi|}.\end{aligned}$$

Corollary 13 (Implicit Key Authentication)**.** *The* IKEYAUTH *predicate is fulfilled for both parties if kem, PRF and sig are secure:*

$$\begin{aligned}&\forall \text{ adversaries } A : Adv^{G_{\text{IKEYAUTH}}}_{A,\Pi,\mathcal{I},\mathcal{I}}(1^\lambda)\\ &\leq l^2 \cdot \left(\frac{4(\varepsilon_{kem}+\varepsilon_{PRF})}{\frac{1}{l^2}-2(\varepsilon_{kem}+\varepsilon_{PRF})-n\varepsilon_{sig}} + 2(\varepsilon_{kem}+\varepsilon_{PRF})\right)\\ &+ 2 \cdot \left(\frac{l^2}{2^{2\lambda}} + l\delta_{kem}\right) + \frac{1}{2^\lambda}\end{aligned}$$

[SFW19] Saint Guilhem, Fischlin, and Warinschi, *Authentication in Key-Exchange: Definitions, Relations and Composition*

iKeyAuth follows from iKeyAuth $\Leftarrow$ iEntAuth $\wedge$ Match $\wedge$ KMSoundness [SFW19, Prop. 9.2.] and from Theorems 4 and 5 and Corollary 12.

Corollary 14 (Full Explicit Key Authentication, cf. Definition 8.2.10)**.** *The* fexKeyAuth *predicate is fulfilled for the air station AS if kem, PRF and sig are secure:*

$$\begin{aligned}&\forall \text{ adversaries} A : Adv^{G_{\text{fexKeyAuth}}}_{A,\Pi,\mathcal{I},\mathcal{L}}(1^\lambda)\\ &\leq l^2 \left(\frac{4(\varepsilon_{kem} + \varepsilon_{PRF})}{\frac{1}{l^2} - 2(\varepsilon_{kem} + \varepsilon_{PRF}) - n\varepsilon_{sig}} + 2(\varepsilon_{kem} + \varepsilon_{PRF}) \right)\\ &+ 2\left(\frac{l^2}{2^{2\lambda}} + l\delta_{kem} \right) + \frac{1}{2^\lambda} + n\varepsilon_{sig}\end{aligned}$$

fexKeyAuth follows from fexKeyAuth $\Leftarrow$ fexEntAuth $\wedge$ Match $\wedge$ KMSoundness [SFW19, Prop. 9.2.] and from Theorem 4 and Corollaries 10 and 12.

Corollary 15 (Almost-Full Explicit Key Authentication, cf. Definition 8.2.11)**.** *The* afexKeyAuth *predicate is fulfilled for the ground station GS if kem, PRF and sig are secure:*

$$\begin{aligned}&\forall \text{ adversaries} A : Adv^{G_{\text{afexKeyAuth}}}_{A,\Pi,\mathcal{I},\mathcal{L}}(1^\lambda)\\ &\leq l^2 \left(\frac{4(\varepsilon_{kem} + \varepsilon_{PRF})}{\frac{1}{l^2} - 2(\varepsilon_{kem} + \varepsilon_{PRF}) - n\varepsilon_{sig}} + 2(\varepsilon_{kem} + \varepsilon_{PRF}) \right)\\ &+ \frac{2l^2}{2^{2\lambda}} + 3l\delta_{kem} + \frac{1}{2^\lambda} + n\varepsilon_{sig}\end{aligned}$$

afexKeyAuth follows from iKeyAuth $\wedge$ afKeyConf $\Leftrightarrow$ afexKeyAuth [SFW19, Thm. 3.1.] and from Theorem 8 and Corollary 13.

9.3 SYMBOLIC SECURITY

The symbolic proof model establishes the correctness and security of a cryptographic protocol without making assumptions about the computational limitations of an adversary. The adversary is modeled as a Dolev-Yao attacker [NS78; DY83], who can send, receive or alter messages in interactive protocols, and who has access to cryptographic primitives as perfect black-boxes. The messages sent over the network are inputs to these black-boxes and the adversary is restricted to use only these. A protocol is formalized using correspondence assertions of the form "If event x is executed, then event y is executed". An automated symbolic prover, such as Tamarin [Mei+13], can check whether a protocol fulfills security goals by (exhaustively) exploring the possible options for parties (called agents) assuming the various roles in the protocol.

[NS78] Needham and Schroeder, "Using Encryption for Authentication in Large Networks of Computers"

[DY83] Dolev and Yao, "On the Security of Public Key Protocols"

[Mei+13] Meier et al., "The TAMARIN Prover For The Symbolic Analysis Of Security Protocols"

In Tamarin [Mei+13], security is formalized over a system state as an initially empty multi-set of predicates called *facts*. *Rules* define how the state can transition to a new set, adding ore removing facts from the system state. Each rule is associated with a premise and a conclusion. A rule can only be applied, if all facts of the premise are present in the system state. If the rule is applied, then the system state is updated according to the facts in

the conclusion. Additionally, rules can feature action facts, which record the application of the rule by appending all action facts to a trace.

With these rules the algorithmic behaviors of a protocol can be modeled by defining rules mimicking the interaction of the agents. Similarly, adversarial powers can be modeled via action facts, for example, a rule can feature an action fact representing that a long-term secret is available to the adversary and that the corresponding party is corrupted.

Security properties on the other hand are modeled via traces of action facts. These are denoted lemmas. This means, a property is modeled by a giving a formula which is evaluated on the traces. Then a security property can be modeled either with an existential ("exists") or an universal ("all") quantification, indicating that the formula has to evaluate to *True* either for any one trace, or for all traces. Usually, security properties are required to hold for all traces.

PROTOCOL AND ADVERSARIAL RULES. The Tamarin manual [The23] already provides built-in code for signing and hashing, Celi et al. [Cel+22, Sec. 3.2.1] provide a Tamarin instantiation of a KEM, which satisfies Definition 2.2.1. The protocol described in Section 9.1 is modeled via rules and facts, allowing each party corresponding to a unique identifier to register a long-term key, and then to engage with any other party in a protocol session.

[The23] The Tamarin Team, *Tamarin-Prover Manual*

[Cel+22] Celi et al., "A Tale of Two Models: Formal Verification of KEMTLS via Tamarin"

To model the adversary additional rules are added that allow to reveal the long-term key, a KEM key or a session key: The rule "Reveal_ltk" has as premises a long-term key associated with an agent X, an action fact that marks the agent as corrupted via the fact "CorruptedLtk(X)", and which adds the long-term key in question as a conclusion fact to the state. Analogously, "Reveal_kem" and "Leak_session" allow to reveal the secret key output by ENCAPS, DECAPS, or the session key. As such, the adversary in the symbolic model is slightly stronger than in the computational model, as they are allowed to have the KEM key leaked (in the computational model, only long-term key via a *Corrupt* query and session keys via *Reveal* queries (or *Test* queries, in the BRSEC predicate)).

SECURITY PROPERTIES. In the following we describe the lemmas associated with the security properties relevant to this work, where agent A and B correspond to the ground- and air stations:

mutual_authentication_A/B The property is modeled via Lowes [Low97] bi-directional full-agreement property as outlined in the Tamarin manual [The23], combined with a unique element exchanged between agent A and B. Full agreement is the combination of injective agreement, i. e., stating that if agent A accepts with agent B, then they can be sure B also accepted with A, except if either of long-term keys was revealed (resulting in corruption of the agent as an action fact). The uniqueness property is modeled via the *session_ uniqueness_A/B* lemma (modeled as in [Gaz+21]), i. e., the guarantee that different sessions have different keys. When both lemmas hold, mutual authentication is achieved via full-agreement. Hence, the *mutual_authentication_A/B* lemma can be regarded as a combination of Definitions 8.2.4 to 8.2.6.

[Low97] Lowe, "A Hierarchy of Authentication Specifications"

[Gaz+21] Gazdag et al., "A Formal Analysis of IKEv2's Post-Quantum Extension"

secrecy Secrecy corresponds to BR-secrecy, promising that when a session key x is assigned a "secret" property at time i, then either the adversary does not know x, or x has been revealed, or a corresponding agent was marked as corrupted via an action fact. Similarly, *secrecy_pfs* guarantees that the adversary does not know x, except if the agent was corrupted at a time j prior to i, i. e., $j < i$.

key_consistency_A/B Key consistency [Gaz+21, Sec. A.6.4] corresponds to key confirmation, such that for all sessions with agents A and B with keys keyA and keyB respectively, the following holds: When agent A accepts keyA at time i in session i_A, and agent B accepts with keyB at time j in the same session, then keyA and keyB must be same, except if one of the agents was corrupted. This corresponds to key confirmation in the computational model.

[Gaz+21] Gazdag et al., "A Formal Analysis of IKEv2's Post-Quantum Extension"

The models of session uniqueness and key confirmation are based on the implementation from Donenfeld et al. [DM17].

[DM17] Donenfeld and Milner, "Formal verification of the WireGuard protocol"

9.3.1 *Transition Model*

Now we are ready to review the automated proof from the Tamarin implementation of the LDACS MAKE protocol as described in Section 9.1. Figure 9.5 outlines the flow of the Tamarin code relative to the description of the protocol in Figure 9.3 as a set of states, rules and protocol messages for each of the agents. Recall that agent A corresponds to the ground station *GS* and agent B to the air station *AS*.

The states of the agents are "S_A_i" for $i \in \mathbb{Z}$ for agent A and "S_B_i" for agent B respectively. The rules, corresponding to the transition between the states of the individual agents (here A) within the protocol, are "$init\_A$", "A_1", "A_2", etc., as well as the queries accessible to the adversary, for example "Reveal_ltk". Additionally, Figure 9.5 shows the Protocol messages implied by the rules, as well as the cryptographic messages applied in Figure 9.3. It may be noted that in the symbolic model the parties first exchange the identifier of the intended partner in plain before engaging in the protocol (in comparison to taking the respective identifier as an input in Figure 9.3).

Our Tamarin code comprises four implementations of the protocol: First, the expected protocol Figure 9.3 with the key encapsulation mechanism. Second, the protocol but excluding the generation, transmission and verification of MACs, to verify that all lemmas can be achieved without using the MAC. The third and fourth variant correspond to the protocol using DH instead of a KEM, thus resembling the original KAM-7 [ISO21, Sec. 11.7] for completeness.

[ISO21] ISO copyright office, *ISO/IEC 11779-3*

For each variant, Tamarin exhaustively explores the possible actions of the agents and the adversary to check if the lemmas *mutual_authentication_A/B*, *session_uniqueness*, *secrecy*, *secrecy_pfs* and *key_consistency_A/B* as described in Section 9.3 are fulfilled. Additionally, the model includes a lemma *session_exists* and *two_sessions_exist*, with the first lemma showing the possibility for the Tamarin model to terminate and the second even enhancing the attacker's abilities by allowing them to reuse cryptographic primitives from the previous protocol run.

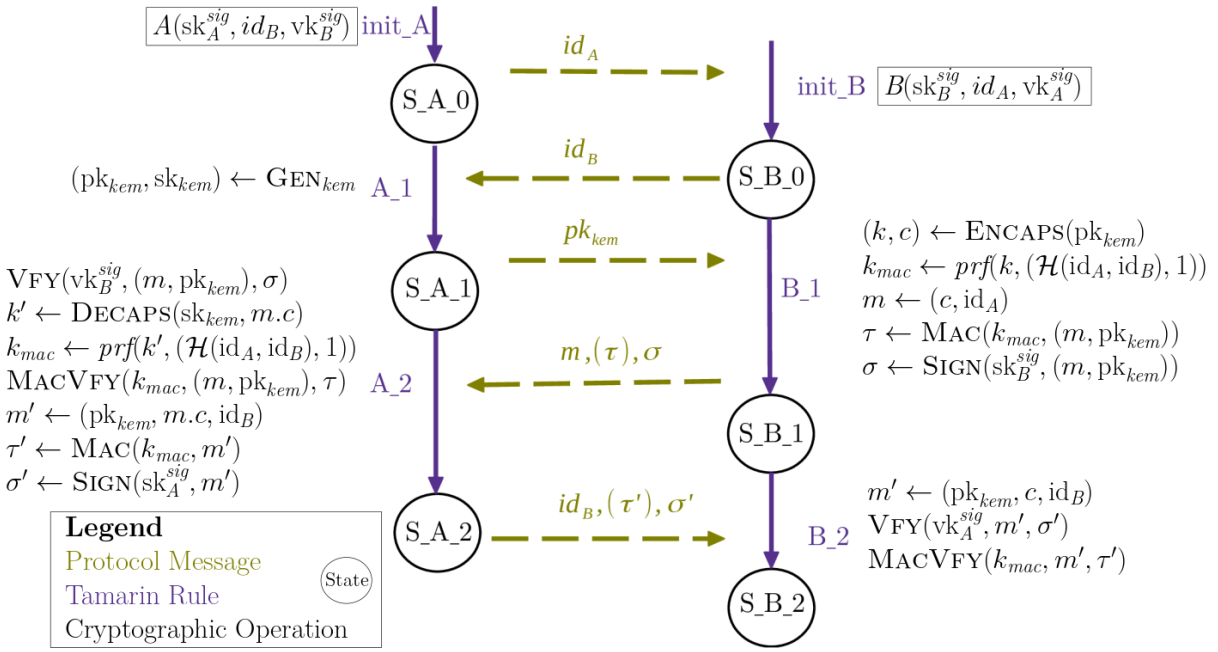


FIGURE 9.5: Tamarin rules, states, cryptographic messages and operations for the KEM variant of the protocol. This figure was inspired by [Cel+22, Fig. 2].

9.3.2 *Results*

Table 9.2 shows the results of verifying the lemmas with Tamarin: the "Scope[-traces]" correspond to either the existence of a sequence of traces ("Exists"), or the exhaustive search over all ("All") possible combinations of traces that achieve the conditions required to fulfill the lemma. The "Steps" indicate the number of rules (see Figure 9.5) that had to be explored to validate the result.

TABLE 9.2: Tamarin results for LDACS security notions. Modeled in-/excluding a MAC and using Diffie–Hellman key exchange/KEM as key exchange.

Lemma	Scope [-traces]	#Steps w HMAC		#Steps w/o HMAC	
		DH	KEM	DH	KEM
session_exists	Exists	24 ✓	25 ✓	23 ✓	23 ✓
two_sessions_exist	Exists	46 ✓	48 ✓	44 ✓	44 ✓
mutual_authentication_A/B	All	50 ✓	54 ✓	50 ✓	54 ✓
session_uniqueness_A/B	All	32 ✓	32 ✓	32 ✓	32 ✓
secrecy	All	32 ✓	28 ✓	24 ✓	26 ✓
secrecy_pfs	All	32 ✓	28 ✓	24 ✓	26 ✓
key_consistency	All	16 ✓	16 ✓	16 ✓	16 ✓

Table 9.2 shows that both lemmas, *session_exists* and *two_sessions_exist*, have a trace, guaranteeing the completion of a protocol run and that the lemmas hold even if the adversary re-uses values from previous runs. Further, for all combinations of setups (i. e., with Diffie–Hellman or KEM, with or without MAC) the lemmas are satisfied, thus the protocol fulfills the security

claims in the symbolic model. Particularly, the lemmas hold even in the setting with the HMAC removed from the protocol, thus the automated proof confirms the findings of Section 9.2, verifying that the message authentication code is not required to provide security for LDACS MAKE in the setting of weak corruption.

10

Making an Asymmetric PAKE Quantum-Annoying

Parts of this chapter are verbatim from our publications [TES23b; TES23a].

A wide-spread method for authentication in client-server situations involves a key exchange where the server is authenticated through a public-key infrastructure, while the client authenticates themselves with a password by transmitting the password directly over the encrypted channel. This method is suboptimal since the user's password is exposed to the server.

A PAKE protocol enables two parties to perform a key exchange, authenticated using mutual knowledge of a shared password, without revealing the password to the network or to each other. The setting of PAKEs allows two kinds of attacks: online attacks (the adversary interacting with either party), and offline attacks (the adversary operating locally based on what is has observed from previous online interactions). Password-based protocols are always vulnerable to online dictionary attacks, where the adversary can rule out one password guess with each online interaction with a party. The goal of a PAKE is to ensure that offline dictionary attacks are infeasible, for example because of an intractability assumption. While PAKEs have been known for decades, there was little progress in adoption for many years, but there is renewed interest in adoption of PAKEs via a variety of recent and ongoing standardization efforts [Sch17; Bou+23; TW22; IEE09; IEC17].

Most PAKEs are based on the hardness of solving the discrete logarithm problem (see [HO22] for an overview), making them vulnerable to attacks by quantum computers, thus motivating the question of building PAKEs that are quantum-resistant. The obvious answer is to build new PAKEs that rely on post-quantum intractability assumptions, and post-quantum PAKEs are starting to emerge in the literature. These new PAKEs (e. g., see [Beg+23]) are based on key encapsulation mechanisms to match the standardized quantum-secure encryption [Nat22]. However, there may be other interim options requiring fewer modifications by augmenting existing protocols.

Quantum Annoying. Eaton and Stebila [ES21] developed a formalization of the quantum-annoying property for PAKEs by considering a classical adversary working in the generic group model who is given the additional power of a discrete logarithm oracle. They showed that the base version of the symmetric PAKE protocol CPace [AHH21] was quantum-annoying in the generic group model. One main characteristic of CPace that lead to it being quantum-annoying is that the password π shared by the client and server is

[Sch17] Schmidt, *Requirements for Password-Authenticated Key Agreement (PAKE) Schemes*

[Bou+23] Bourdrez et al., *The OPAQUE Asymmetric PAKE Protocol*

[TW22] Taubert and Wood, *SPAKE2+, an Augmented PAKE*

[IEE09] IEEE, *IEEE Standard Specification for Password-Based Public-Key Cryptographic Techniques*

[IEC17] IEC, *Information technology – Personal identification – ISO-compliant driving licence*

[HO22] Hao and Oorschot, "SoK: Password-Authenticated Key Exchange – Theory, Practice, Standardization and Real-World Lessons"

[Beg+23] Beguinet et al., "GeT A CAKE: Generic Transformations From Key Encapsulation Mechanisms To Password Authenticated Key Exchanges"

[Nat22] National Institute for Standards and Technology, *NIST: Selected Algorithms 2022*

[ES21] Eaton and Stebila, "The "Quantum Annoying" Property of Password-Authenticated Key Exchange Protocols"

[AHH21] Abdalla, Haase, and Hesse, "Security Analysis of CPace"

used to derive a generator g_π of the group, and then a Diffie–Hellman key exchange is performed using that generator (g_π^{xy}). But from the perspective of an adversary who only sees Diffie–Hellman public keys (g_π^x and g_π^y), no information is gained about the password π since for each π' there is an x' such that $g_{\pi'}^{x'} = g_\pi^x$.

CONTRIBUTION & OBJECTIVE. In this manuscript, we focus on KHAPE [GJK21], a compiler that turns a Key-Hiding Authenticated Key Exchange (KH-AKE) and a PAKE into an asymmetric Password Authenticated Key Exchange (aPAKE). Asymmetric PAKEs improve upon regular PAKEs by forcing an attacker to perform an exhaustive search on the password even after server compromise, since the value stored by the server cannot be used to impersonate the client. The OPAQUE framework [JKX18] introduced the notion of *strong* asymmetric PAKEs, which further guarantees that no pre-computation can be performed to aid in the exhaustive search for the password in the case of server compromise. This is achieved by combining an oblivious pseudo-random function and a PAKE.

[GJK21] Gu, Jarecki, and Krawczyk, "KHAPE: Asymmetric PAKE from Key-Hiding Key Exchange"

[JKX18] Jarecki, Krawczyk, and Xu, "OPAQUE: An Asymmetric PAKE Protocol Secure Against Pre-computation Attacks"

Whereas CPace is a symmetric PAKE, the KHAPE-HMQV protocol constructed by the KHAPE compiler [GJK21] is an asymmetric PAKE, so compromise of a server using KHAPE-HMQV does not enable the adversary to impersonate a user without first performing an offline dictionary attack. However, the protocol is not quantum-annoying: after seeing just a single transcript, a single discrete logarithm computation suffices to enable an offline dictionary attack to recover the user's password. We address this vulnerability by presenting the QA-KHAPE protocol, a quantum-annoying variant of KHAPE-HMQV. As shown in Figure 10.1, our modifications entail encapsulating an additional key into the server-stored credentials, which is later used by the principals to encrypt their Diffie–Hellman key-pairs prior to exchanging messages. This effectively means that each guess of the password causes the transcript to decrypt (under a symmetric key dependent on the password) to a different pair of Diffie–Hellman public keys, so a new discrete logarithm must be performed each time.

The changes to the protocol require only minimal computational and communication overhead, with the same number of rounds as KHAPE-HMQV and only a single additional ideal cipher ciphertext (increasing the server-client ciphertext from three to four elements). The client-server communication remains unchanged, and the protocol requires two additional ideal cipher computations, one encryption, and one decryption.

We show that QA-KHAPE is quantum-annoying following the methodology of [ES21]: the adversary is a classical adversary in the generic group model with the addition of a discrete logarithm oracle. In Section 10.2.1, we define a security game in the generic group model tailored to capturing the core quantum annoying property of the QA-KHAPE protocol. In Section 10.3, we apply this to show that QA-KHAPE is secure in a quantum-annoying variant of the standard BPR00 security model (cf. Section 8.3) for asymmetric password authenticated key exchange.

[ES21] Eaton and Stebila, "The "Quantum Annoying" Property of Password-Authenticated Key Exchange Protocols"

LIMITATIONS. Just as in the original security proof of KHAPE by [GJK21], our analysis also relies on the ideal cipher assumption. Care must be taken for an instantiation of the IC, which is discussed in [GJK21, Sec. 8].

In 2023, a preprint (later published as [HYY24]) has examined the "multiple discrete logarithm" problem induced by the quantum annoying model [HYY23]. In this work, the authors show that it is possible to (asymptotically) solve m discrete logarithm problems (in a generic group) with a quantum computer more efficiently than m times the cost of a single Shor's instance. In particular, their algorithm solves m discrete logarithms with around $\log m$ times fewer quantum group operations (if $m = \Omega(\log p)$, where p is the size of the group). This comes at the expense of requiring large quantum memory to compute everything simultaneously. Whether this represents a concrete improvement to the ability of an adversary to break quantum annoying security (and if so, how large the grouping m should be) is an interesting open question. Our proofs bound the adversary's success probability in terms of the number of discrete logarithm oracle queries made. If it is a practical improvement to group such queries, this does not affect our proofs, only how the induced bounds translate to real-world estimates of adversary cost.

[HYY24] Hhan, Yamakawa, and Yun, "Quantum Complexity for Discrete Logarithms and Related Problems"

[HYY23] Hhan, Yamakawa, and Yun, *Quantum Complexity for Discrete Logarithms and Related Problems*

10.1 QUANTUM ANNOYING KHAPE-HMQV

The KHAPE compiler [GJK21] transforms a KH-AKE, a PAKE, a random oracle, and an ideal cipher into an asymmetric PAKE which provides key establishment with key integrity and confirmation, mutual authentication and forward secrecy. A highly efficient instantiation [GJK21, Fig. 14] uses the HMQV [Kra05] protocol, the security of which is based on the computational Diffie–Hellman problem.

[GJK21] Gu, Jarecki, and Krawczyk, "KHAPE: Asymmetric PAKE from Key-Hiding Key Exchange"

[Kra05] Krawczyk, "HMQV: A High-Performance Secure Diffie-Hellman Protocol"

KHAPE is split into a *registration* and an aPAKE phase. During registration the server generates the KH-AKE key pairs $(a, A := g^a)$, $(b, B := g^b)$, partially encrypts them using the password as a key, $e \leftarrow \mathrm{IC}.E(\pi, a, B)$, and stores the ciphertext along with (A, b). All other values are discarded. In the aPAKE phase the server generates a key pair (y, Y) and sends (Y, e) to the client. The client decrypts e using their password and generates a key pair (x, X). A Diffie–Hellman session is computed from (a, x, B, Y) which is used to derive a key-confirmation value τ, and later the session key. The key confirmation is sent along with the value X to the server, who computes the equivalent Diffie–Hellman session from (b, y, A, X), verifies the key confirmation, and either computes a session key and a new key confirmation (which is send to the client), or sets both to $\perp$. The client checks the key confirmation and computes the session key, or sets it to $\perp$.

In the quantum-annoying setting, KHAPE-HMQV is susceptible to an offline attack on the password using a single discrete logarithm query. Given a transcript (e, Y, X, τ) an adversary can determine a list of possible values for KH-AKE key pairs: each password guess π_i gives a pair of candidate values $(a_i, B_i) \leftarrow \mathrm{IC}.D(\pi_i, e)$. Additionally, they can query the discrete logarithm oracle once on the value X, receiving x. Then for each password guess (i. e., for each a_i, B_i), they can verify if the Diffie–Hellman completion results in the key-confirmation value τ from the transcript, effectively providing an offline method to check passwords.

10.1.1 *QA-KHAPE*

Our QA-KHAPE protocol, presented in Figure 10.1, is a quantum-annoying aPAKE. The construction is based on KHAPE-HMQV and requires only minimal changes, which are highlighted in the figure. During the *registration* phase the server generates an additional secret key *sk* which is then encrypted using the π and stored as part of the credentials. Correspondingly, during the *aPAKE* phase the client decrypts e obtaining this key *sk*, which they use to encrypt the ephemeral value X, resulting in the ciphertext *c*, which is then sent to the server. Briefly speaking, QA-KHAPE is quantum-annoying because an adversary receiving a transcript must now solve a discrete logarithm for *every* decryption of *c* or e to verify if a password guess was correct. This comes at the cost of an additional secret key to be stored as credentials, which increases the size of first message from server to client. The client has to perform one additional decryption and encryption, while the server only performs an additional decryption.

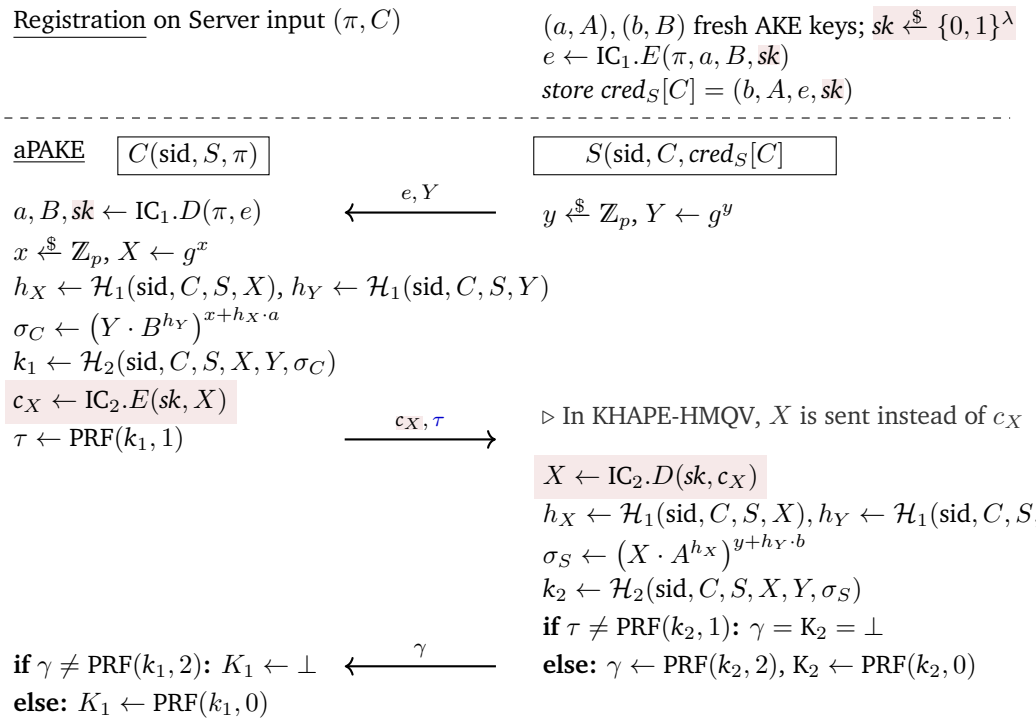


FIGURE 10.1: QA-KHAPE: quantum-annoying variant of KHAPE-HMQV [GJK21, Fig 14], with our changes compared to KHAPE-HMQV highlighted.

SECURITY. The QA-KHAPE protocol is a quantum-annoying aPAKE in the generic group (cf. Section 8.4), ideal cipher and random oracle model and features mutual authentication and key confirmation. No perfect forward secrecy can be achieved in the setting of quantum-annoying for KHAPE-HMQV, because compromise of any party releases a static secret that, together with the public value e, removes all the ambiguity on the group elements in question (i. e., A, B, X). This enables an offline attack on the password using only a single discrete logarithm query. Note that a quantum-annoying PAKE achieving perfect forward secrecy would mean to establish a secure, authenticated key without taking advantage of the password or credentials, which

seemingly contradicts the main point of a PAKE; establishing this formally is an interesting question for future work.

All other properties of KHAPE-HMQV are preserved, for example, security based on the computational Diffie–Hellman assumption against purely classical attackers, and thus a full fall back to security of KHAPE-HMQV. The quantum-annoying security is summarized in our main contribution, Theorem 10.

Theorem 10. *Let $\mathbb{G}$ be a cyclic group of size p, $\mathcal{H}_1, \mathcal{H}_2$ be random oracles and IC_1, IC_2 ideal ciphers with ciphertext space $\{0,1\}^{n_1}$, $\{0,1\}^{n_2}$ respectively. Let $q_{\text{SEND}}, q_{\text{EXEC}}, q_{\mathcal{H}_i}, q_{IC_i}, q_\circ, q_{\text{DLOG}}$ be the number of queries to the QA-BPR oracles, and let ϵ_{PRF} an adversary's chance to distinguish PRF from a random function. Let N be the size of the password space for π. Then the advantage of an adversary to win the QA-BPR game for the QA-KHAPE protocol in Figure 10.1 is bounded by*

$$Adv^{\text{QA-KHAPE}}_{\text{QA-BPR}} \leq \frac{q_{\text{DLOG}} + q_{\text{SEND}}}{N} + \epsilon$$

$$\epsilon := \frac{q_{\text{EXEC}} + q_{\text{SEND}}}{\epsilon_{PRF}^{-1}} + \frac{(q_{IC_1} + q_{IC_2} + q_\circ)^2 + (q_{\text{DLOG}} q_\circ^2)}{p} + \frac{q_{\text{EXEC}}}{2^{n_1}} + \frac{q_{\text{EXEC}} + q_{\text{SEND}}}{2^{n_2}}$$

$$+ \frac{q_{\text{SEND}} \cdot (q_\circ + 1)}{p} + \frac{(2q_{IC_1} + q_{IC_2})}{p} + \frac{(q_{IC_1})}{2^\kappa} + \frac{(2q_{IC_1}^2 + q_{IC_2}^2)}{p} + \frac{(q_{IC_1}^2)}{2^\kappa} + \frac{q_{\mathcal{H}_2}}{p}$$

We prove Theorem 10 in two steps: first, in Section 10.2.1, we introduce the KHAPE$_{\text{CORE}}$-game that captures the quantum-annoying property of QA-KHAPE in the generic group model. Briefly speaking, the game models the aPAKE without key-confirmation values and is defined such that any adversary can only win if they query the *correct* Diffie–Hellman completion to the random oracle. This allows us to quantify the number of discrete logarithm queries required, and to prove that every password guess requires either an online interaction, or a respective discrete logarithm query. Formally, this is captured in Theorem 11 which we prove in Section 10.2.2. Second, we reduce the QA-BPR-security of the QA-KHAPE protocol to the KHAPE$_{\text{CORE}}$-game, which is represented by Theorem 10 and which we prove in Section 10.3. Together, these yield the proof of the quantum-annoying property.

10.2 SECURITY FRAMEWORK: THE KHAPE$_{\text{CORE}}$ GAME

We define a game KHAPE$_{\text{CORE}}$ that captures the quantum annoying property of the protocol in Figure 10.1, namely the indistinguishability of the keys k_1, k_2 from random, which translates the approach of [ES21, Sec 3] into the setting of an aPAKE.

[ES21] Eaton and Stebila, "The "Quantum Annoying" Property of Password-Authenticated Key Exchange Protocols"

10.2.1 *Security Game*

The game is defined over a set $[L]$ of registrants; each $l \in [L]$ is associated with static, secret variables π_l, sk_l, a_l, B_l and a static, public variable e_l. The variables are set on initialization of the KHAPE$_{\text{CORE}}$-game via the REGISTRATION oracle (cf. Figure 10.2), along with uniformly random sampled

challenge bit s. Additionally, each registrant l is associated with a counter ctr_l initialized to 0 corresponding to the interaction with the lth set of static variables. Each interaction is called an instance. The adversary may interact with an arbitrary number of registrants and instances through a set of oracles, eventually allowing the adversary to obtain the keys k_1, k_2. The challenge bit determines if these keys are real (if $s = 0$), in which case they are computed from Diffie–Hellman session, or random (if $s = 1$).

INTERFACE. The oracles take as input a value l matching a set of static variables which are used by the game to respond to a query. Ephemeral variables for an instance (l, ctr_l) are stored for consistent use by the other oracles. The PASSIVEEXEC oracle (cf. Figure 10.2) corresponds to a passive execution of the protocol in Figure 10.1, excluding the key confirmation values. The ACTIVE$_C$ or ACTIVE$_S$ oracles (cf. Figure 10.3), correspond to interacting with, or impersonating, either party in the QA-KHAPE protocol, and thus at most one of the two may be queried for each instance. The ACTIVE oracles compute, depending on the value of the challenge bit s, either a key value $k_{l,ctr_l,i}$ from the input and the static variables or output a uniformly random string. The GETSTATIC oracle (cf. Figure 10.2) mimics the corruption of parties, which causes the game to reprogram the outputs of the ACTIVE oracles into the respective positions before it returns the secret static variables. Finally, the adversary is given access to the random oracles $\mathcal{H}_1, \mathcal{H}_2$, the block-ciphers IC_1, IC_2 modeled by ideal ciphers and access to an interface of the generic group model.

REGISTRATION(l)

$\pi_l \xleftarrow{\$} [N]$, $sk_l \xleftarrow{\$} \{0,1\}^\lambda$
$a_l, b_l \xleftarrow{\$} \mathbb{Z}_p$
$B_l \leftarrow g^{b_l}$, $A_l \leftarrow g^{a_l}$
$e_l \leftarrow \text{IC}_1.E(\pi_l, a_l, B_l, sk_l)$
Store $\pi_l, sk_l, e_l, A_l, b_l$

PASSIVEEXEC(l)

Get stored $\pi_l, sk_l, e_l, A_l, b_l$
Increment ctr_l
$x_{l,ctr_l}, y_{l,ctr_l} \xleftarrow{\$} \mathbb{Z}_p$
$X_{l,ctr_l} \leftarrow g^{x_{l,ctr_l}}$, $Y_{l,ctr_l} \leftarrow g^{y_{l,ctr_l}}$
$c_{l,ctr_l} \leftarrow \text{IC}_2.E(\pi_l, X_{l,ctr_l})$
Store $Y_{l,ctr_l}, ctxt_{l,ctr_l}$
return $e_l, Y_{l,ctr_l}, c_{l,ctr_l}$

GETSTATIC(l)

Mark l corrupted
Get stored π_l, sk_l
for $m = 0, \ldots, ctr_l$
  for $k_{l,m}, c_{l,m} \leftarrow$ ACTIVE$_C(l, e, Y)$
    $a, B, sk \leftarrow \text{IC}_1.D(\pi_l, e)$
    $h_X = \mathcal{H}_1(l, m, X_{l,m})$, $h_Y = \mathcal{H}_1(l, m, Y)$
    $\sigma_C \leftarrow (Y \circ B^{h_Y})^{x_{l,m}+h_X \cdot a}$
    $\mathcal{H}_2(l, m, X_{l,m}, Y, \sigma_C) := k_{l,m,1}$
  for $k_{l,2} \leftarrow$ ACTIVE$_S(l, c)$
    Get stored $Y_{l,m}$
    $X \leftarrow \text{IC}_2.D(sk_l, c)$
    $h_X = \mathcal{H}_1(l, m, X)$, $h_Y = \mathcal{H}_1(l, m, Y_{l,m})$
    $\sigma_S \leftarrow (X \circ A_{l,m}^{h_X})^{y_{l,m}+h_Y \cdot b_{l,m}}$
    $\mathcal{H}_2(l, m, X, Y_{l,m}, \sigma_S) := k_{l,m,2}$
return (π_l, sk_l)

FIGURE 10.2: KHAPE$_{\text{CORE}}$ algorithms.

OUTPUT. The KHAPE$_{\text{CORE}}$-game outputs 1 if the adversary's output matches the challenge bit s or if they if they query $\mathcal{H}_2(l, m, X, Y, \sigma_{l,C})$ (respectively $\mathcal{H}_2(l, m, X, Y, \sigma_{l,S})$) after submitting a query ACTIVE$_C(l, e, Y)$ (respectively ACTIVE$_S(l, c)$), but before querying GETSTATIC(l) on the instance. The adversary is then said to *win* the game. The restriction on the GETSTATIC oracle mimics the fact that forward secrecy cannot be achieved in the quantum annoying model. The conditions under which the game outputs 1 are analogous to the winning conditions of [ES21, Sec. 3.1].

[ES21] Eaton and Stebila, "The "Quantum Annoying" Property of Password-Authenticated Key Exchange Protocols"

$\textsc{Active}_S(l, c)$

1 : Increment ctr_l; Get stored sk_l
2 : **if** challenge $s = 0$ or l corrupted
3 : $\quad X \leftarrow \text{IC}_2.D(sk_l, c)$
4 : $\quad h_X = \mathcal{H}_1(l, ctr_l, X)$, $h_Y = \mathcal{H}_1(l, ctr_l, Y_{l,ctr_l})$
5 : $\quad \sigma_C \leftarrow (X \circ A_l^{h_X})^{y_{l,ctr_l} + h_Y \cdot b_{l,ctr_l}}$
6 : $\quad k_{l,ctr_l,2} = \mathcal{H}_2(l, ctr_l, X, Y_{l,ctr_l}, \sigma_{l,C})$
7 : **else**
8 : $\quad k_{l,ctr_l,2} \leftarrow \{0,1\}^n$
9 : **return** $k_{l,ctr_l,2}$

$\textsc{Active}_C(l, e, Y)$

1 : Increment ctr_l; Get stored π_l
2 : $a, B, sk \leftarrow \text{IC}_1.D(\pi_l, e)$
3 : $x_{l,ctr_l} \xleftarrow{\$} \mathbb{Z}_p$, $X_{l,ctr_l} \leftarrow g^{x_{l,ctr_l}}$
4 : $c_{l,ctr_l} \leftarrow \text{IC}_2.E(sk, X_{l,ctr_l})$
5 : **if** challenge $s = 0$ or l corrupted
6 : $\quad h_X = \mathcal{H}_1(l, ctr_l, X_{l,ctr_l})$, $h_Y = \mathcal{H}_1(l, ctr_l, Y)$
7 : $\quad \sigma_C \leftarrow (Y \circ B^{h_Y})^{x_{l,ctr_l} + h_X \cdot a}$
8 : $\quad k_{l,ctr_l,1} = \mathcal{H}_2(l, ctr_l, X_{l,ctr_l}, Y, \sigma_C)$
9 : **else**
10 : $\quad k_{l,ctr_l,1} \leftarrow \{0,1\}^n$
11 : **return** $k_{l,ctr_l,1}, c_{l,ctr_l}$

FIGURE 10.3: KHAPE$_{\text{CORE}}$ algorithms

10.2.2 *Security of KHAPE$_{CORE}$*

Now we are ready to define Theorem 11 which states, informally, that the adversary's chance to win the KHAPE$_{\text{CORE}}$-game is limited by their ability to query the DLOG oracle on the *correct* group element, or any of the ACTIVE oracles on a ciphertext encoding a group element the discrete logarithm of which is known to them.

Theorem 11. *Let q_{AE_C}, q_{AE_S} be the number of queries to the ACTIVE and q_{PE} the number of queries to the PASSIVEEXEC oracle. Let $q_{IC_i}, q_{\mathcal{H}_i}, q_\circ, q_{DLOG}$ be the number of queries to the ideal cipher, random oracle, group operation and discrete logarithm oracles respectively. Then an adversary's probability to win the KHAPE$_{CORE}$-game is bounded by*

$$\mathbb{P}\left[KHAPE_{CORE} \to 1\right] \leq \frac{1}{2} + \frac{q_{AE_C} + q_{AE_S} + q_{DLOG}}{N} + \epsilon_{CORE}$$

$$\epsilon_{CORE} := \overbrace{\frac{(q_{IC_1} + q_{IC_2} + q_\circ)^2 + (q_{DLOG} q_\circ^2)}{p}}^{// \, G_0 \rightsquigarrow G_1} + \overbrace{\frac{(q_{AE_C} + q_{AE_S}) \cdot (q_\circ + 1)}{p}}^{// \, G_3 \rightsquigarrow G_4}$$

$$+ \underbrace{\frac{q_{PE}}{2^{n_1}} + \frac{q_{PE} + q_{AE}}{2^{n_2}}}_{// \, G_2 \rightsquigarrow G_3} + \underbrace{\frac{(2q_{IC_1} + q_{IC_2})}{p} + \frac{(q_{IC_1})}{2^\kappa} + \frac{(2q_{IC_1}^2 + q_{IC_2}^2)}{p} + \frac{(q_{IC_1}^2)}{2^\kappa}}_{// \, G_1 \rightsquigarrow G_2} .$$

Proof. In the KHAPE$_{\text{CORE}}$-game, the group elements in question are computed as

$$\begin{aligned}\sigma_C &= \left(Y \cdot B^{h_Y}\right)^{x + h_X \cdot a} = Y^x \cdot Y^{h_x \cdot a} \cdot B^{h_Y \cdot x} \cdot B^{h_y \cdot h_x \cdot a} \\ &= g^{xy} \cdot g^{h_X \cdot a \cdot y} \cdot g^{h_Y \cdot b \cdot x} \cdot g^{h_Y \cdot h_X \cdot a \cdot b} \\ &= X^y \cdot A^{h_X \cdot y} \cdot X^{h_Y \cdot b} \cdot A^{h_X \cdot h_Y \cdot b} = \left(X \cdot A^{h_X}\right)^{y + h_Y \cdot b} = \sigma_S \,,\end{aligned}$$

where computing σ_C, σ_S depends on either the knowledge of $\text{DLOG}(g, B)$ or $\text{DLOG}(g, X)$. The framework presented in Section 8.4, allows us to quantify if these element are knowable based on the number of discrete logarithm queries. This is possible, because the relevant group elements X, B are encrypted under the ideal cipher. On a decryption query the ideal cipher can return a public representations that does not admit a relation to a previously received group element known by the adversary. To learn any such relation, the adversary then has to query the DLOG oracle. Specifically, the *relevant*

group elements $\{B_{l,i}, X_{l,i}\}_{i\in[N]}$ correspond to decryptions of (e, c) using a password guess π_i and sk_i as keys respectively. In the KHAPE$_{\text{CORE}}$-game, the *correct* pair $B_{l,i}, X_{l,i}$ is chosen during the *Registration* phase and in the ACTIVE oracles. Due to the values being encrypted by the ideal ciphers, the simulation does not need to commit to any actual pair B_i, X_i.

We prove this by presenting a sequence of game hops where the initial game G_0 (cf. Section 10.2.2) is the KHAPE$_{\text{CORE}}$-game as defined in Section 10.2.1, and G_4 (cf. Section 10.2.2) is modified such that the keys k_1, k_2 are chosen uniformly random for every instance, and where the discrete logarithm of g and the group elements B, X remain undefined unless sufficiently constrained by queries to the DLOG and ACTIVE oracles. They are undefined because the ciphertexts (e, c) are indistinguishable from random strings, and the key pair (π, sk) is no longer defined from the PASSIVEEXEC or ACTIVE oracles. That means that the *correct* values for (B, X) may correspond to any of the N possible pairs. As long as there is a degree of freedom left for these representations, the discrete logarithm relative to g is also not defined, and the random oracle cannot be queried on the respective Diffie–Hellman completion. These are only defined either if an instance is corrupted, or if sufficiently many discrete logarithms have been queried, allowing to quantify the adversary's probability to win relative to the number of DLOG queries.

G_0 (KHAPE$_{\text{CORE}}$-GAME). This is the KHAPE$_{\text{CORE}}$ as described in Section 10.2.1.

G_1 (GGM). We modify the responses to the group operation $\circ$ and DLOG oracle by simulating the generic group as described in Section 8.4. The generator initially given to the adversary is $g_1 = g$, which corresponds to the secret representation 1. The secret representation of the neutral element is 0. Recall that the password space is of size N. The secret variables are represented as a set $\{\chi_{l,i}, \chi_{l,i}\}_{i\in[N]}$ corresponding to the pairs $B_{l,i}, X_{l,i}$ that can be obtained when querying the ideal ciphers on possible values for π_l or sk_l. The ideal cipher IC_i is maintained via a table T_{IC_i}. On query $\text{IC}_1.D(\pi, e)$, if $T_{\text{IC}_1}[\pi, e]$ is defined, return $T_{\text{IC}_1}[\pi, e]$. Otherwise, sample a random index $j \xleftarrow{\$} [N]$ for the secret representation and a public representation $g_V \xleftarrow{\$} \{0,1\}^n$, both of which are added to the table $T_{\text{ggm}}[\chi_{i,j}] := g_V$; The public representation g_V is returned. The simulation of IC_2 is analog.

The modification changes the distribution of the group elements: public representations returned from the ideal ciphers (on new inputs) in the simulation are unique, whereas the adversary would expect a collision after $\sqrt{p}$ new queries. Additionally, the adversary would expect to see collisions between random public representations, and the elements returned from (sufficiently many) group operations. This happens with probability $(q_{\text{IC}_1} + q_{\text{IC}_2} + q_\circ)^2/p$.

Further, a group element may be assigned two distinct public representations, if first computed from group operations and then returned from an IC query (or vice versa). For example, if the public representation g_x was returned from an IC query, and the representation $g_x = g^x$ was assigned from group operations, then the adversary may detect the modification by computing $\text{DLOG}(g_1, g_x) = x$. The probability that this happens for group elements randomly assigned by the ideal cipher and for all DLOG queries is $q_{\text{DLOG}} q_\circ^2/p$.

REGISTRATION(l)	PASSIVEEXEC(l)
1 : $e_l \xleftarrow{\$} \{0,1\}^{n_1}$	1 : Get stored e_l; Increment ctr_l
2 : Store e_l	2 : $c_{ctr,X} \xleftarrow{\$} \{0,1\}^{n_2}$
	3 : $y_{ctr} \leftarrow \mathbb{Z}_p$, $Y_{ctr} \leftarrow g^{y_{ctr}}$
	4 : **return** $(Y_{ctr}, e_{ctr}), (c_{ctr,X})$

FIGURE 10.4: Simulation in G_3.

Overall, the adversary can distinguish the two games with probability at most

$$\frac{(q_{\text{IC}_1} + q_{\text{IC}_2} + q_\circ)^2 + q_{\text{DLOG}} q_\circ^2}{p}.$$

G_2 (IDEAL CIPHERS OUTPUT). We change the ideal ciphers to output unique, random values when queried on a new input. On query $\text{IC}_1.D(\pi, e)$, if $T_{\text{IC}_1}[\pi, e]$ is not defined, the ideal cipher IC_1 samples key pairs $a, b \xleftarrow{\$} \mathbb{Z}_p$ and $sk \leftarrow \{0,1\}^\lambda$, generates public keys $A = g^a$, $B = g^b$ and a key $sk \leftarrow \{0,1\}^\lambda$, and programs $T_{\text{IC}_1}[\pi, e] := a, B, sk$. In the case of a collision, i. e., if (a, B, sk) has been assigned to a value in the map $T_{\text{IC}_1}[\pi, \cdot]$ for any value $\cdot$, G_2 aborts. Since (a, B, sk) are independent random variables, the probability for an abort is bounded by $2q_{\text{IC}_1}/p + q_{\text{IC}_1}/2^\kappa$, neglecting a deduction for a simultaneous collision of all variables. Since the values a, B, sk are *unique*, two different queries will never output the same values, whereas the adversary would eventually expect a collision in G_1. The same argument applies to IC_2. In total, the divergence is bounded by

$$\frac{2q_{\text{IC}_1} + q_{\text{IC}_2}}{p} + \frac{(q_{\text{IC}_1})}{2^\kappa} + \frac{2q_{\text{IC}_1}^2 + q_{\text{IC}_2}^2}{p} + \frac{(q_{\text{IC}_1}^2)}{2^\kappa}.$$

G_3 (RANDOM CIPHERTEXTS). We modify the game to not sample any keys π and sk and to output random strings $e \xleftarrow{\$} \{0,1\}^{n_1}$, $c \xleftarrow{\$} \{0,1\}^{n_2}$ in the PASSIVEEXEC and ACTIVE$_C$ oracles, which removes the game's commitment to any value stored in (e, c). This change is explicitely presented in the simulation of REGISTRATION(l) in Figure 10.4 and of PASSIVEEXEC(l) in Figure 10.4. Analogous to the modification in G_2, the game aborts if the values were previously assigned.

The modification of the ideal cipher as in G_2 is not reflected in the Figure 10.4.

At the same time the GETSTATIC oracle is changed to reflect this modification as depicted in Figure 10.5: the simulation first decrypts the ciphertext using freshly sampled keys π and sk. The DLOG oracle provides the values necessary to compute the Diffie–Hellman session such that the output of the ACTIVE oracles can be programmed into the correct position of the ideal cipher.

The distribution of (e, c) returned by PASSIVEEXEC and ACTIVE$_C$ is the same as in G_2 unless it aborts. Since (e, c) are sampled uniformly random from the ciphertext space, the probability for this to happen is bounded by

$$\frac{q_{\text{PE}}}{2^{n_1}} + \frac{q_{\text{PE}} + q_{\text{AE}_C}}{2^{n_2}}.$$

G_4 (EMBED RANDOM KEYS). The ACTIVE oracles are modified to always return random strings $k \xleftarrow{\$} \{0,1\}^\lambda$ for non-corrupted instances. To notice

GETSTATIC(l)

1 : Mark l corrupted
2 : $\pi_l \leftarrow [N], sk_l \leftarrow \{0,1\}^\lambda$
3 : **for** $m = 0, \ldots, ctr_l$
4 : **for** $k_{l,m}, c_{l,m} \leftarrow$ ACTIVE$_C(l, e, Y)$
5 : $a, B, sk \leftarrow \mathsf{IC}_1.D(\pi_l, e)$
6 : $X \leftarrow \mathsf{IC}_2.D(sk, c_{l,m})$; $x \leftarrow$ DLOG(g, X)
7 : $h_X = \mathcal{H}_1(l, m, X)$, $h_Y = \mathcal{H}_1(l, m, Y)$
8 : $\sigma_{l,m,C} \leftarrow (Y \circ B^{h_Y})^{x + h_X \cdot a}$
9 : $\mathcal{H}_2(l, m, X, Y, \sigma_{l,m,C}) := k_{l,m,1}$
10 : **for** $k_{l,2} \leftarrow$ ACTIVE$_S(l, c)$
11 : $X \leftarrow \mathsf{IC}_2.D(sk_l, c)$
12 : $a, B, sk_2 \leftarrow \mathsf{IC}_1.D(\pi_l, e_l)$; $b \leftarrow$ DLOG(g, B)
13 : $h_X = \mathcal{H}_1(m, l, X)$, $h_Y = \mathcal{H}_1(m, l, Y_{m,l})$
14 : $\sigma_{m,l,C} \leftarrow (X \circ A^{h_X})^{y_l + h_Y \cdot b}$
15 : $\mathcal{H}_2(m, l, X, Y_{m,l}, \sigma_{m,l,S}) := k_{m,l,2}$
16 : **return** (π_l, sk_l)

FIGURE 10.5: Simulation of GETSTATIC(l) in G_3.

this change, the adversary must query $\mathcal{H}_2(ctr, X, Y, \sigma_i)$, where the Diffie–Hellman completion σ_i depends on either the knowledge of DLOG(g, X) and B, or the knowledge of DLOG(g, B) and X, both of which are not defined by the game unless GETSTATIC has been queried, in which case the adversary cannot win the game anymore.

The probability that DLOG(g, X) or DLOG(g, B) are knowable to the adversary is bounded by Corollary 9, which tells us that the discrete logarithms are defined if an only if $\vec{b}, \vec{x}$ are in the row span of D. Both, $\vec{b}$ and $\vec{x}$, are basis vectors with a 1 at the position of the random index associated with the respective secret variable. The number of basis vectors that can appear in the row span are upper bounded by the rank of the matrix D, which is increased by 1 for each DLOG query. Therefore, the probability that the adversary can force the definition for any one value out of N of these is bounded by q_{DLOG}/N.

Remark: Only public representations returned from the ideal ciphers, and possibly group elements that come from group operation applied to these group elements, provide *useful* input to the DLOG oracle, since the discrete logarithm relation for group elements originating purely from g is already known to the adversary. Therefore, the probability is

$$\frac{\min(q_{\mathsf{IC}_1} + q_{\mathsf{IC}_2}, q_{\text{DLOG}})}{N} \leq \frac{q_{\text{DLOG}}}{N}\,.$$

Additionally, the adversary may submit a query with a group element, the discrete logarithm of which is known to them. The input $\hat{e}$ to the ACTIVE$_C$ oracle is either a value formerly returned from a previous query to PASSIVE-EXEC, in which case the adversary must also query the ideal cipher and the DLOG oracle and there is a chance of $(q_\circ + 1)/p$ that the discrete logarithm of the group element decrypted by ACTIVE$_C$ is known to them. If $\hat{e}$ was crafted by the adversary, i. e., if they queried the ideal cipher on values $\hat{a}, \hat{B}, \hat{sk}$ such that the discrete logarithm of $\hat{B}$ is known to them, then they expect an $1/N$ chance that there choice of $\hat{pw}$ was correct, and that ACTIVE$_C$ used $\hat{b}$ to compute the Diffie–Hellman session.

In total, this result in a divergence for ACTIVE_C queries bounded by

$$\frac{q_{\text{AE}_C} \cdot (q_\circ + 1)}{p} + \frac{q_{\text{AE}_C}}{N}.$$

For ACTIVE_S, the adversary may submit a value $\hat{c_x}$ for which the same arguments hold, resulting in a total probability for either of both occurring of

$$\frac{(q_{\text{AE}_C} + q_{\text{AE}_S}) \cdot (q_\circ + 1)}{p} + \frac{\left(q_{\text{AE}_C} + q_{\text{AE}_S}\right)}{N}.$$

Finally, the adversary's advantage to distinguish the simulation from the real game based on DLOG queries depends on the knowledge of at least one key from a ACTIVE oracle, resulting in a factor of $\min(q_{\text{AE}_C} + q_{\text{AE}_S}, 1)$, thus bounding the overall divergence by

$$\begin{aligned}
&\min(q_{\text{AE}_C} + q_{\text{AE}_S}, 1) \cdot \\
&\left(\frac{(q_{\text{AE}_C} + q_{\text{AE}_S}) \cdot (q_\circ + 1)}{p} + \frac{\left(q_{\text{AE}_C} + q_{\text{AE}_S} + \min(q_{\text{IC}_1} + q_{\text{IC}_2}, q_{\text{DLOG}})\right)}{N}\right) \\
&\qquad \leq \frac{(q_{\text{AE}_C} + q_{\text{AE}_S}) \cdot (q_\circ + 1)}{p} + \frac{\left(q_{\text{AE}_C} + q_{\text{AE}_S} + q_{\text{DLOG}}\right)}{N}
\end{aligned}$$

In G_4, the PASSIVEEXEC and ACTIVE_C oracle output random values as ciphertexts e, c that do not commit to any values a, B, π or X. Particularly, the values $\text{DLOG}(g, X), \text{DLOG}(g, B)$ are defined only upon corruption or after a number of ACTIVE and DLOG queries relative to the password space N. The ACTIVE_* oracles further output a random key independent of the challenge bit $s = 0$. The adversary is left with either guessing the challenge bit, or querying values to $\mathcal{H}_2$. This concludes the proof of Theorem 11.

□

10.3 PROOF OF APAKE SECURITY

The security of the QA-KHAPE protocol (cf. Figure 10.1) is proven in the QA-BPR (cf. Section 8.3) model. Recall that the adversary may interact through the EXECUTE, SEND, REVEAL, CORRUPT and TEST oracles after the *Registration* phase, where the protocol defines how the principals respond. Additionally, the adversary has access to the group operation, DLOG and random oracle, ideal cipher and pseudo-random function PRF, as described in Section 8.4. Here we prove, that the adversary is bounded as stated in Theorem 10.

Proof. We consider a sequence of games starting with G_0, which corresponds to the QA-KHAPE protocol illustrated in Figure 10.1. As we progress to G_3, the sessions keys are chosen independent and uniformly at random, ensuring that the adversary A is reduced to a simple guessing attack. Throughout this reduction process, we present an adversary $\mathcal{B}$ on the $\text{KHAPE}_{\text{CORE}}$-game, which maintains a mapping between instances of the $\text{KHAPE}_{\text{CORE}}$-game and instances of principals in the QA-BPR-model. The oracles provided by the $\text{KHAPE}_{\text{CORE}}$ challenger are referred to as $\text{KHAPE}_{\text{CORE}}$.Oracle. Throughout the sequence of game hope we utilize a procedure called *CoreMap*, which bears resemblance to the GETUV procedure described in [ES21, App. B.2].

[ES21] Eaton and Stebila, "The "Quantum Annoying" Property of Password-Authenticated Key Exchange Protocols"

CoreMap. We define a function *CoreMap*(C, S, sid) which maps parties and their respective sessions in the QA-BPR model onto the KHAPE$_{\text{CORE}}$. Recall that a session in the QA-BPR model is denoted as $\ell = (i, j, k)$, where i, j are parties and k indicates that this is the k-th session between i and j. Further, each such session is associated with a *unique* string sid. In the following, we consider the parties C and S instead of i and j.

The function *CoreMap* defines the counter $\bar{l}$, which maps parties in the QA-BPR model to an instance of the KHAPE$_{\text{CORE}}$. Similarly, the counters and $ctr_{\bar{l}}$, $ctr_{C,S}$, $ctr_{C,S,\text{sid}}$ corresponding to individual sessions of this party. The mapping is required, because each party must have a consistent view on the output of the generic group model oracles, which are internally simulated using static variables.

All counters are initialized to *zero*, and incremented throughout the simulation. The *CoreMap* works as follows:

- If the $ctr_{C,S,\text{sid}} > 0$, the respective transcript $e_{ctr_{C,S}}, Y_{ctr_{C,S},ctr_{C,S,\text{sid}}}, \mathsf{c}_{ctr_{C,S},ctr_{C,S,\text{sid}}}$ has been generated previously and is returned.
- Otherwise, if $ctr_{C,S} = 0$, then this is the first interaction with party l. The reduction sets $ctr_{C,S} \leftarrow \bar{l}$, $ctr_{\bar{l}} \leftarrow 1$, increments $\bar{l}$, corresponding to ctr_l in the KHAPE$_{\text{CORE}}$, and sets $ctr_{C,S,\text{sid}} \leftarrow 1$. The oracle KHAPE$_{\text{CORE}}$.PassiveExec$(ctr_{C,S})$ is queried; the output stored and returned.
- If $ctr_{C,S} > 0$, The reduction sets $ctr_{C,S,\text{sid}} \leftarrow ctr_{\bar{l}}$, increments $ctr_{\bar{l}}$ and queries KHAPE$_{\text{CORE}}$.PassiveExec$(ctr_{C,S})$. The output is stored in $e_{ctr_{C,S}}, Y_{ctr_{C,S},ctr_{C,S,\text{sid}}}, \mathsf{c}_{ctr_{C,S},ctr_{C,S,\text{sid}}}$ and returned.

G_0 (Figure 10.1). This is the real protocol.

G_1 (Passive Sessions). The game is modified by replacing the keys k_1, k_2 with random values for passively observed sessions. Particularly, on input Execute(C, S, sid), we set $k_1 = k_2 \leftarrow \{0,1\}^\lambda$ and compute the key confirmation values τ, γ and sessions keys using the PRF. The adversaries oracle calls to all instances l for which Execute has been called are simulated as follows: First, the simulation invokes *CoreMap*(C, S, sid) to obtain k_1, c'_X from KHAPE$_{\text{CORE}}$.Active$_C(ctr_{C,S,\text{sid}}, e, Y)$, is used to compute the confirmation values τ, γ. On a Corrupt(C, S) query, the extraction calls KHAPE$_{\text{CORE}}$.GetStatic$(ctr_{C,S})$ returning π_l, sk_l, which programs the key k_1 returned by a KHAPE$_{\text{CORE}}$.Active oracle into the *correct* position of the random oracle $\mathcal{H}_2$. The extraction receives a, B, sk from the ideal cipher on query IC$_1$.$D(sk_1, e)$ as well as the discrete logarithm b from KHAPE$_{\text{CORE}}$.Dlog(B). It then computes $A \leftarrow g^a$. Let T_π be a table corresponding to all N passwords. The extraction sets $\pi \leftarrow \mathcal{P}[sk_1]$, i. e., the sk_1'th entry of the table and returns $\pi, (e, A, b, sk)$, which is a perfect simulation.

For the queries $\mathcal{H}_1(\text{sid}, C, S, *)$, if the entry $ctr_{C,S,\text{sid}}$ is defined, the query is forwarded to the KHAPE$_{\text{CORE}}$-challenger, and the result is returned. Otherwise, a random value is sampled uniformly at random from the range of $\mathcal{H}_1$, and a table is maintained for consistent responses. $\mathcal{H}_2(\text{sid}, C, S, *)$ is simulated analogous to $\mathcal{H}_1$. All queries to IC$_1$ and IC$_2$ are forwarded to the KHAPE$_{\text{CORE}}$-challenger. In Section 10.3 the divergence $q_{\mathcal{H}_2}/p$ from this

simulation, i. e., the random oracle and ideal cipher queries, has already been discussed.

Finally, the adversary may query a TEST or REVEAL query, receiving the session key from the KHAPE$_{\text{CORE}}$-game, In the first case, if a TEST query has been received,

extraction either simulates either G_0, if the KHAPE$_{\text{CORE}}$ challenge bit is *zero*, or G_1, if the KHAPE$_{\text{CORE}}$ challenge bit s is *one*. When $s = 0$, the values of e, c_X as well as τ, γ are distributed as expected (i. e., as in G_0), since the keys $k_1 = k_2$ are identical and thus γ can also be computed from k_1. On the other hand, if $s = 1$, the key k_1 is chosen uniformly random as expected, and thus the key confirmation values also have the expected distribution.
In the second case, if a REVEAL query has been received, key k_1 returned from the simulation is real-or-random, but would be expected to always be real, resulting in a divergence.

However, from an adversary detecting this change an extraction of a winning query to the KHAPE$_{\text{CORE}}$ can be provided: In order to notice the change, the adversary A has to query the random oracle on $\mathcal{H}_2(\text{sid}, C, S, X, Y, \sigma_C)$ or $\mathcal{H}_2(\text{sid}, C, S, X, Y, \sigma_S)$, both of which allow to instantly win the KHAPE$_{\text{CORE}}$-game. Note that the key confirmation values returned by the aPAKE impact the advantage to win the KHAPE$_{\text{CORE}}$, since even a passive execution allows to verify if a derived session key is correct. Therefore, the term $\min(q_{\text{AE}_C} + q_{\text{AE}_S}, 1)$ is 1. Further, the inputs to *CoreMap*.ACTIVE$_*$ are sampled in KHAPE$_{\text{CORE}}$.PASSIVEEXEC such that no *new* group elements, the discrete logarithm of which is knowable to the adversary, have to be considered in the probability to win the KHAPE$_{\text{CORE}}$-game. Consequently, the number of these queries is exactly the number of EXECUTE queries. The probability to detect the difference between game G_0 and G_1 is then bounded by

$$\frac{q_{\text{DLOG}}}{N} + \epsilon_{\textit{passiv}} + \frac{q_{\mathcal{H}_2}}{p}$$

$$\epsilon_{\textit{passiv}} := \frac{(q_{\text{IC}_1} + q_{\text{IC}_2} + q_\circ)^2 + (q_{\text{DLOG}} q_\circ^2)}{p} + \frac{q_{\text{IC}_1}^2 + q_{\text{EXECUTE}}}{2^{n_1}} + \frac{q_{\text{IC}_2}^2 + q_{\text{EXECUTE}}}{2^{n_2}}.$$

G_2 (ACTIVE SESSIONS). In G_2, the modifications (i. e., replacing k_1, k_2 with random strings),are extended to active sessions:

- For an active session impersonating a client C, the oracle calls are modified as follows: On input SEND$(C, l, M = (S, \text{sid}))$ the simulation responds with the values e, Y retrieved from KHAPE$_{\text{CORE}}$.PASSIVEEXEC. On input SEND$(C, l, M = (S, \text{sid}, c_X, \tau))$ we sample the $k_2 \leftarrow \{0,1\}^\lambda$ uniformly at random and computes $\tau' \leftarrow \text{PRF}(k_2, 1)$. The session key and the key confirmation value are generated from k_1, k_2 based on $\tau = \tau'$ as in a genuine execution of the protocol.

- For an active session impersonating a server S, the oracle calls are modified as follows: On input SEND$(C, l, M = (S, \text{sid}, e, Y))$ the simulation samples a uniformly random value for $k_1 \leftarrow \{0,1\}^\lambda$ and computes the key confirmation value τ using the PRF. On input SEND$(C, l, M = (S, \text{sid}, \gamma))$ we compute $\gamma' \leftarrow \text{PRF}(k_1, 2)$ and set the session key conditionally on the outcome of $\gamma = \gamma'$ (i. e., as in the *real* protocol).

On queries to the random oracle, ideal cipher, REVEAL and CORRUPT the reduction behaves identical to G_1, and thus the divergence is identical. Eventually, the adversary may query a TEST or REVEAL query receiving a session key from the $\text{KHAPE}_{\text{CORE}}$. To bound the adversaries chance to detect the modification, a similar extractor of a winning query to the $\text{KHAPE}_{\text{CORE}}$-game is provided. Similarly to Section 10.3, the reduction calls *CoreMap* to map instances of the QA-BPR-game to instances of the $\text{KHAPE}_{\text{CORE}}$-game. The extraction of a winning query on the adversary SEND queries to clients and servers is examined separately.

Impersonation of clients: On $\text{SEND}(C, l, M = (S, \text{sid}))$ the extraction calls $\textit{CoreMap}(C, S, \text{sid})$, which causes $\textit{ctr}_{C,S}$ to become defined if it previously was not, and the retrieved values e, Y are returned. On $\text{SEND}(C, i, M = (S, \text{sid}, c_X, \tau))$ the reduction calls $\textit{CoreMap}(C, S, \text{sid})$ to subsequently obtain $k_2 \leftarrow \text{KHAPE}_{\text{CORE}}.\text{ACTIVE}(\textit{ctr}_{C,S}, c_X)$. The key confirmation value τ' is computed from the obtained key using the PRF. The session key and key confirmation value are set conditioned on $\tau = \tau'$ as in the real protocol.

Impersonation of Server: On $\text{SEND}(S, i, M = (C, \text{sid}, j, e, Y))$, the reduction calls $\textit{CoreMap}(C, S, \text{sid})$, which causes $\textit{ctr}_{C,S}$ to become defined if it previously was not. Then the reduction calls $k_1 \leftarrow \text{KHAPE}_{\text{CORE}}.\text{ACTIVE}(\textit{ctr}_{C,S}, e, Y)$ and computes the key confirmation value τ genuinely using the PRF, and returns c_X, τ. On $\text{SEND}(S, i, M = (C, j, \gamma, \text{sid}))$, the reduction computes γ' from the key k_2 using the PRF and compares this to γ. If they match, the session key is set to $\text{K}_1 \leftarrow \text{PRF}(k_1, 0)$, and otherwise, to $\perp$. For SEND the arguments are analogous to G_1: If TEST was queried, the reduction simulates G_1 (and thus G_0) perfectly if the $\text{KHAPE}_{\text{CORE}}$ challenge bit $s = 0$, and simulates G_2 if $s = 1$(except for inconsistencies in the random oracle). Otherwise, the adversary can detect the change only by querying the random oracle on either of the two inputs $\mathcal{H}_2(\text{sid}, C, S, X, Y, \sigma_C)$ or $\mathcal{H}_2(\text{sid}, C, S, X, Y, \sigma_S)$, both of which are winning queries for the reduction in $\text{KHAPE}_{\text{CORE}}$.

The number of ACTIVE queries for which the adversary may choose the input is bounded by the number of SEND queries, bounding the difference between game G_1 and G_2 by

$$\frac{(q_{\text{DLOG}} + q_{\text{SEND}})}{N} + \epsilon_{\textit{active}} + \frac{q_{\mathcal{H}_2}}{p},$$

with

$$\epsilon_{\textit{active}} := \frac{(q_{\text{IC}_1} + q_{\text{IC}_2} + q_\circ)^2 + (q_{\text{DLOG}} q_\circ^2)}{p} + \frac{q_{\text{IC}_1}^2 + q_{\text{SEND}}}{2^{n_1}} + \frac{q_{\text{IC}_2}^2 + q_{\text{SEND}}}{2^{n_2}}.$$

G_3 (RANDOM SESSIONS KEYS). The final modification in G_3 (i. e., replacing the session keys with random strings) was discussed in Section 10.3, resulting in the term $(q_{\text{EXEC}} + q_{\text{SEND}})\epsilon_{\text{PRF}}$. The sessions keys are now uniformly random and independent of the password and credentials leaving adversary to a guessing attack.

The probability that the adversary can distinguish G_0 from G_3 is bounded by

$$\frac{(q_{\text{DLOG}} + q_{\text{SEND}})}{N} + \frac{q_{\mathcal{H}_2}}{p} + (q_{\text{EXEC}} + q_{\text{SEND}})\epsilon_{\text{PRF}} + \epsilon,$$

with

$$\epsilon \leq \frac{(q_{\mathrm{IC}_1} + q_{\mathrm{IC}_2} + q_{\circ})^2 + (q_{\mathrm{DLOG}} q_{\circ}^2)}{p} + \frac{(q_{\mathrm{IC}_1}^2 + q_{\mathrm{SEND}} + q_{\mathrm{EXEC}})}{2^{n_1}} + \frac{(q_{\mathrm{IC}_2}^2 + q_{\mathrm{SEND}} + q_{\mathrm{EXEC}})}{2^{n_2}}.$$

This conclude the proof. □

Conclusion

We have demonstrated the security achievable in higher-level protocols when secure post-quantum cryptography is employed, as well as security that can be achieved even without post-quantum assumptions. At the same time, the finding show that secure post-quantum cryptographic primitives can be integrated into higher-level protocols to ensure their resilience against quantum adversaries.

At first, our findings reveal that robust security be maintained for the mutual authentication deployed in the LDACS air-to-ground communication protocol when switching over to quantum-secure primitives. Specifically we have shown that LDACS achieves Entity Authentication, capturing that the promise that each party is assured that they interact with their intended peer, and that their peer is aware of the identity of the party. Further, we have shown that Key Authentication holds, giving the parties assurance that their intended peer and only their intended peer knows the secret key. Both properties hold when the signature is instantiated with a post-quantum, EUF-CMA signature, and when the KEM is instantiated with a post-quantum IND-CPA KEM.

Secondly, our research highlights the potential of enhancing existing protocols, ensuring robust security in the face of emerging quantum threats. This enhancement is based on intractability assumptions that remain difficult for classical computers while being solvable by quantum computers. Surprisingly, the enhancement allow to achieve a notion of quantum-security.

The common point across Part II and Part III is that while *new* quantum-secure cryptographic schemes may bring with them significant vulnerabilities, cryptography is "in good shape" overall. That means, it appears that one can construct quantum-secure schemes from primitives such a hash functions and lattices which can be deployed in higher level protocols without loosing important properties. On the other side, even if those scheme will not be deemed fully secure, we may still have hope for *some* level of security that allows to scale the resources an adversary has to spend in order to compromise cryptographic protocols.

Bibliography

[Aar23] Scott Aaronson. *Introduction to Quantum Information Science Lecture Notes*. 2023. URL: https://www.scottaaronson.com/qclec.pdf (cit. on p. 31).

[AA03] Scott Aaronson and Andris Ambainis. "Quantum Search of Spatial Regions". In: 2003, pp. 200–209. DOI: 10.1109/SFCS.2003.1238194 (cit. on pp. 109, 110).

[AHH21] Michel Abdalla, Björn Haase, and Julia Hesse. "Security Analysis of CPace". In: *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 IV*. Ed. by Mehdi Tibouchi and Huaxiong Wang. Vol. 13093. Lecture Notes in Computer Science. Springer, 2021, pp. 711–741. DOI: 10.1007/978-3-030-92068-5_24. URL: https://doi.org/10.1007/978-3-030-92068-5%5C_24 (cit. on p. 175).

[Aer21] Aeronautical Radio, Incorporated (ARINC). *Internet Protocol Suite (IPS) for Aeronautical Safety Services Part 1 Airborne IPS System Technical Requirements*. ARINC SPECIFICATION 858P1. June 2021 (cit. on pp. 155, 160).

[Agg+17a] Divesh Aggarwal, Antoine Joux, Anupam Prakash, and Miklos Santha. *A New Public-Key Cryptosystem via Mersenne Numbers*. Cryptology ePrint Archive, Report 2017/481. https://eprint.iacr.org/2017/481. 2017 (cit. on pp. 10, 28, 47).

[Agg+17b] Divesh Aggarwal, Antoine Joux, Anupam Prakash, and Mikos Santha. *Mersenne-756839*. Technical report, National Institute of Standards and Technology. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-1-submissions. 2017 (cit. on pp. 10, 28, 47–49, 65–67).

[Ajt96] Miklós Ajtai. "Generating hard instances of lattice problems (extended abstract)". In: *Electron. Colloquium Comput. Complex.* TR96 (1996). URL: https://api.semanticscholar.org/CorpusID:6864824 (cit. on p. 59).

[AKS01] Miklós Ajtai, Ravi Kumar, and D. Sivakumar. "A sieve algorithm for the shortest lattice vector problem". In: 2001, pp. 601–610. DOI: 10.1145/380752.380857 (cit. on pp. 61, 105).

[Alb+20a] Martin R. Albrecht, Shi Bai, Pierre-Alain Fouque, Paul Kirchner, Damien Stehlé, and Weiqiang Wen. "Faster Enumeration-Based Lattice Reduction: Root Hermite Factor $k^{1/(2k)}$ Time $k^{k/8+o(k)}$". In: 2020, pp. 186–212. DOI: 10.1007/978-3-030-56880-1_7 (cit. on pp. 61, 62, 105, 119).

[Alb+21] Martin R. Albrecht, Shi Bai, Jianwei Li, and Joe Rowell. "Lattice Reduction with Approximate Enumeration Oracles - Practical Algorithms and Concrete Performance". In: 2021, pp. 732–759. DOI: 10.1007/978-3-030-84245-1_25 (cit. on pp. 61, 105).

[Alb+18] Martin R. Albrecht, Benjamin R. Curtis, Amit Deo, Alex Davidson, Rachel Player, Eamonn W. Postlethwaite, Fernando Virdia, and Thomas Wunderer. "Estimate All the LWE, NTRU Schemes!" In: 2018, pp. 351–367. DOI: 10.1007/978-3-319-98113-0_19 (cit. on pp. 61, 105, 124).

[Alb+19a] Martin R. Albrecht, Léo Ducas, Gottfried Herold, Elena Kirshanova, Eamonn W. Postlethwaite, and Marc Stevens. "The General Sieve Kernel and New Records in Lattice Reduction". In: 2019, pp. 717–746. DOI: 10.1007/978-3-030-17656-3_25 (cit. on pp. 61, 105).

[Alb+19b] Martin R. Albrecht, Vlad Gheorghiu, Eamonn W. Postlethwaite, and John M. Schanck. *Estimating quantum speedups for lattice sieves*. Cryptology ePrint Archive, Paper 2019/1161. https://eprint.iacr.org/2019/1161. 2019. URL: https://eprint.iacr.org/2019/1161 (cit. on p. 40).

[Alb+20b] Martin R. Albrecht, Vlad Gheorghiu, Eamonn W. Postlethwaite, and John M. Schanck. "Estimating Quantum Speedups for Lattice Sieves". In: 2020, pp. 583–613. DOI: 10.1007/978-3-030-64834-3_20 (cit. on pp. 12, 41, 64, 106).

[Alb+17] Martin R. Albrecht, Florian Göpfert, Fernando Virdia, and Thomas Wunderer. *Revisiting the Expected Cost of Solving uSVP and Applications to LWE*. Cryptology ePrint Archive, Report 2017/815. https://eprint.iacr.org/2017/815. 2017 (cit. on p. 60).

[APS15a] Martin R. Albrecht, Rachel Player, and Sam Scott. *On The Concrete Hardness Of Learning With Errors*. Cryptology ePrint Archive, Report 2015/046. https://eprint.iacr.org/2015/046. 2015 (cit. on p. 60).

[APS15b] Martin R. Albrecht, Rachel Player, and Sam Scott. "On the concrete hardness of Learning with Errors". In: *Journal of Mathematical Cryptology* 9.3 (2015), pp. 169–203. DOI: doi:10.1515/jmc-2015-0016. URL: https://doi.org/10.1515/jmc-2015-0016 (cit. on pp. 105, 113, 125).

[Ali21] Dante Alighieri. *Divine Comedy*. 1321 (cit. on p. 3).

[Alk+16] Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. "Post-quantum Key Exchange - A New Hope". In: 2016, pp. 327–343 (cit. on pp. 61, 105).

[AK17] Andris Ambainis and Martins Kokainis. "Quantum algorithm for tree size estimation, with applications to backtracking and 2-player games". In: *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*. Ed. by Hamed Hatami, Pierre McKenzie, and Valerie King. ACM, 2017, pp. 989–1002. DOI: 10.1145/3055399.3055444. URL: https://doi.org/10.1145/3055399.3055444 (cit. on pp. 12, 37, 105).

[Amy+16] Matthew Amy, Olivia Di Matteo, Vlad Gheorghiu, Michele Mosca, Alex Parent, and John M. Schanck. "Estimating the Cost of Generic Quantum Pre-image Attacks on SHA-2 and SHA-3". In: 2016, pp. 317–337. DOI: 10.1007/978-3-319-69453-5_18 (cit. on pp. 11, 42, 43, 58, 84, 87, 99–102).

[Ano24] Anonymous. "Quantum Disadvantage". In: *SIGBOVIK 2024*. 2024, pp. 199–205. URL: https://www.sigbovik.org/2024/proceedings.pdf (cit. on p. 6).

[Aon+18] Yoshinori Aono, Phong Q. Nguyen, Takenobu Seito, and Junji Shikata. "Lower Bounds on Lattice Enumeration with Extreme Pruning". In: 2018, pp. 608–637. DOI: 10.1007/978-3-319-96881-0_21 (cit. on pp. 64, 113, 119, 125, 127, 131).

[ANS18] Yoshinori Aono, Phong Q. Nguyen, and Yixin Shen. "Quantum Lattice Enumeration and Tweaking Discrete Pruning". In: 2018, pp. 405–434. DOI: 10.1007/978-3-030-03326-2_14 (cit. on pp. 12, 61, 64, 105, 106, 108, 110, 113, 117, 125).

[Aon+16] Yoshinori Aono, Yuntao Wang, Takuya Hayashi, and Tsuyoshi Takagi. "Improved Progressive BKZ Algorithms and Their Precise Cost Estimation by Sharp Simulator". In: 2016, pp. 789–819. DOI: 10.1007/978-3-662-49890-3_30 (cit. on p. 61).

[Aru+19] Frank Arute et al. "Quantum supremacy using a programmable superconducting processor". In: *Nature* 574.7779 (Sept. 2019), pp. 505–510. ISSN: 1476-4687. DOI: 10.1038/s41586-019-1666-5. URL: https://doi.org/10.1038/s41586-019-1666-5 (cit. on p. 6).

[BG14] Shi Bai and Steven D Galbraith. "Lattice decoding attacks on binary LWE". In: *Information Security and Privacy: 19th Australasian Conference, ACISP 2014, Wollongong, NSW, Australia, July 7-9, 2014. Proceedings 19*. Springer. 2014, pp. 322–337 (cit. on p. 124).

[Bai+23] Shi Bai, Maya-Iggy van Hoof, Floyd B. Johnson, Tanja Lange, and Tran Ngo. "Concrete Analysis of Quantum Lattice Enumeration". In: *Advances in Cryptology - ASIACRYPT 2023*. Lecture Notes in Computer Science. Springer-Verlag, 2023 (cit. on pp. 41, 106, 136).

[Bal+18] Marco Baldi, Alessandro Barenghi, Franco Chiaraluce, Gerardo Pelosi, and Paolo Santini. *LEDAcrypt: Low-dEnsity parity-chck coDe-bAsed cryptographic systems*. Technical report, National Institute of Standards and Technology. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-2-submissions. 2018 (cit. on p. 65).

[Bec+16] Anja Becker, Léo Ducas, Nicolas Gama, and Thijs Laarhoven. "New directions in nearest neighbor searching with applications to lattice sieving". In: *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*. Ed. by Robert Krauthgamer. SIAM, 2016, pp. 10–24. DOI: 10.1137/1.9781611974331.ch2. URL: https://doi.org/10.1137/1.9781611974331.ch2 (cit. on pp. 61, 105).

[Beg+23] Hugo Beguinet, Céline Chevalier, David Pointcheval, Thomas Ricosset, and Mélissa Rossi. "GeT A CAKE: Generic Transformations From Key Encaspulation Mechanisms To Password Authenticated Key Exchanges". In: Kyoto, Japan: Springer-Verlag, 2023, pp. 516–538. ISBN: 978-3-031-33490-0. DOI: 10.1007/978-3-031-33491-7_19. URL: https://doi.org/10.1007/978-3-031-33491-7_19 (cit. on p. 175).

[Bel06] Mihir Bellare. "New Proofs for NMAC and HMAC: Security without collision-resistance". In: *Advances in Cryptology - CRYPTO 2006, 26th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2006, Proceedings*. Ed. by Cynthia Dwork. Vol. 4117. Lecture Notes in Computer Science. Springer, 2006, pp. 602–619. DOI: 10.1007/11818175_36. URL: https://doi.org/10.1007/11818175%5C_36 (cit. on p. 162).

[BHK09] Mihir Bellare, Dennis Hofheinz, and Eike Kiltz. "Subtleties in the Definition of IND-CCA: When and How Should Challenge-Decryption be Disallowed?" In: *IACR Cryptol. ePrint Arch.* (2009), p. 418. URL: http://eprint.iacr.org/2009/418 (cit. on p. 21).

[BHK15] Mihir Bellare, Dennis Hofheinz, and Eike Kiltz. "Subtleties in the Definition of IND-CCA: When and How Should Challenge Decryption Be Disallowed?" In: *J. Cryptol.* 28.1 (2015), pp. 29–48. DOI: 10.1007/S00145-013-9167-4. URL: https://doi.org/10.1007/s00145-013-9167-4 (cit. on p. 21).

[BPR00] Mihir Bellare, David Pointcheval, and Phillip Rogaway. “Authenticated Key Exchange Secure against Dictionary Attacks”. In: *Advances in Cryptology - EUROCRYPT 2000, International Conference on the Theory and Application of Cryptographic Techniques, Bruges, Belgium, May 14-18, 2000, Proceeding*. Ed. by Bart Preneel. Vol. 1807. Lecture Notes in Computer Science. Springer, 2000, pp. 139–155. DOI: 10.1007/3-540-45539-6_11. URL: https://doi.org/10.1007/3-540-45539-6%5C_11 (cit. on pp. 148, 158).

[Bel13] Aleksandrs Belovs. *Quantum Walks and Electric Networks*. 2013. arXiv: 1302.3143 [quant-ph] (cit. on p. 35).

[Ben89] Charles H. Bennett. “Time/Space Trade-Offs for Reversible Computation”. In: *SIAM Journal on Computing* 18.4 (1989), pp. 766–776. DOI: 10.1137/0218053. eprint: https://doi.org/10.1137/0218053. URL: https://doi.org/10.1137/0218053 (cit. on pp. 32, 38).

[Ben23] Huck Bennett. “The Complexity of the Shortest Vector Problem”. In: *SIGACT News* 54.1 (Mar. 2023), pp. 37–61. ISSN: 0163-5700. DOI: 10.1145/3586165.3586172. URL: https://doi.org/10.1145/3586165.3586172 (cit. on p. 60).

[BT21a] Robin M. Berger and Marcel Tiepelt. “On Forging SPHINCS$^+$-Haraka Signatures on a Fault-Tolerant Quantum Computer”. In: *Progress in Cryptology - LATINCRYPT 2021 - 7th International Conference on Cryptology and Information Security in Latin America, Bogotá, Colombia, October 6-8, 2021, Proceedings*. Ed. by Patrick Longa and Carla Ràfols. Vol. 12912. Lecture Notes in Computer Science. Springer, 2021, pp. 44–63. DOI: 10.1007/978-3-030-88238-9_3. URL: https://doi.org/10.1007/978-3-030-88238-9_3 (cit. on pp. 12, 17, 29, 31, 47, 83, 98, 99).

[BT21b] Robin M. Berger and Marcel Tiepelt. *On Forging SPHINCS+-Haraka Signatures on a Fault-tolerant Quantum Computer*. Cryptology ePrint Archive, Paper 2021/1484. https://eprint.iacr.org/2021/1484. 2021. DOI: 10.1007/978-3-030-88238-9_3. URL: https://eprint.iacr.org/2021/1484 (cit. on pp. 12, 17, 31, 47, 83).

[Ber+17] Daniel J. Bernstein, Nadia Heninger, Paul Lou, and Luke Valenta. *Post-quantum RSA*. Cryptology ePrint Archive, Paper 2017/351. https://eprint.iacr.org/2017/351. 2017. URL: https://eprint.iacr.org/2017/351 (cit. on p. 7).

[Ber+14] Daniel J. Bernstein, Daira Hopwood, Andreas Hülsing, Tanja Lange, Ruben Niederhagen, Louiza Papachristodoulou, Peter Schwabe, and Zooko Wilcox-O’Hearn. “SPHINCS: practical stateless hash-based signatures”. In: *IACR Cryptol. ePrint Arch.* (2014), p. 795. URL: http://eprint.iacr.org/2014/795 (cit. on p. 51).

[Ber+11] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. *Cryptographic sponge functions*. 2011. URL: https://keccak.team/sponge_duplex.html (cit. on pp. 59, 102).

[Bes05] Arvid J Bessen. “Lower bound for quantum phase estimation”. In: *Physical Review A* 71.4 (2005), p. 042313 (cit. on p. 110).

[Beu+19] Marc Beunardeau, Aisling Connolly, Rémi Géraud, and David Naccache. “On the Hardness of the Mersenne Low Hamming Ratio Assumption”. In: 2019, pp. 166–174. DOI: 10.1007/978-3-030-25283-0_9 (cit. on pp. 29, 49–51, 66).

[Bin+23] Nina Bindel, Xavier Bonnetain, Marcel Tiepelt, and Fernando Virdia. *Quantum Lattice Enumeration in Limited Depth*. Cryptology ePrint Archive, Paper 2023/1423. https://eprint.iacr.org/2023/1423. 2023. URL: https://eprint.iacr.org/2023/1423 (cit. on pp. 13, 17, 31, 36, 41, 47, 64, 105, 107–109, 116–119, 125–127, 131, 132).

[Bin+24] Nina Bindel, Xavier Bonnetain, Marcel Tiepelt, and Fernando Virdia. “Quantum Lattice Enumeration in Limited Depth”. In: *Advances in Cryptology – CRYPTO 2024*. Ed. by Leonid Reyzin and Douglas Stebila. Cham: Springer Nature Switzerland, 2024, pp. 72–106. ISBN: 978-3-031-68391-6. DOI: 10.1007/978-3-031-68391-6_3. URL: https://doi.org/10.1007/978-3-031-68391-6_3 (cit. on pp. 13, 17, 30, 31, 47, 105).

[Bin+19] Nina Bindel, Jacqueline Brendel, Marc Fischlin, Brian Goncalves, and Douglas Stebila. “Hybrid Key Encapsulation Mechanisms and Authenticated Key Exchange”. In: *Post-Quantum Cryptography - 10th International Conference, PQCrypto 2019, Chongqing, China, May 8-10, 2019 Revised Selected Papers*. Ed. by Jintai Ding and Rainer Steinwandt. Vol. 11505. Lecture Notes in Computer Science. Springer, 2019, pp. 206–226. DOI: 10.1007/978-3-030-25510-7_12. URL: https://doi.org/10.1007/978-3-030-25510-7%5C_12 (cit. on pp. 157, 162).

[Boe+21] Franziska Boenisch, Reinhard Munz, Marcel Tiepelt, Simon Hanisch, Christiane Kuhn, and Paul Francis. “Side-Channel Attacks on Query-Based Data Anonymization”. In: *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*. CCS ’21. Virtual Event, Republic of Korea: Association for Computing Machinery, 2021, pp. 1254–1265. ISBN: 9781450384544. DOI: 10.1145/3460120.3484751. URL: https://doi.org/10.1145/3460120.3484751 (cit. on p. 16).

[Boe+18] Koen de Boer, Léo Ducas, Stacey Jeffery, and Ronald de Wolf. "Attacks on the AJPS Mersenne-Based Cryptosystem". In: 2018, pp. 101–120. DOI: 10.1007/978-3-319-79063-3_5 (cit. on pp. 49, 51, 61).

[BS23] Dan Boneh and Victor Shoup. *A Graduate Course in Applied Cryptography*. 2023. URL: https://toc.cryptobook.us/ (cit. on pp. 21–23).

[Bon+23] Xavier Bonnetain, André Chailloux, André Schrottenloher, and Yixin Shen. "Finding Many Collisions via Reusable Quantum Walks - Application to Lattice Sieving". In: *Advances in Cryptology - EUROCRYPT 2023 - 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23-27, 2023, Proceedings, Part V*. Ed. by Carmit Hazay and Martijn Stam. Vol. 14008. Lecture Notes in Computer Science. Springer, 2023, pp. 221–251. DOI: 10.1007/978-3-031-30589-4_8. URL: https://doi.org/10.1007/978-3-031-30589-4%5C_8 (cit. on p. 105).

[Bos+17] Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M Schanck, Peter Schwabe, and Damien Stehlé. *CRYSTALS – Kyber: a CCA-secure module-lattice-based KEM*. Cryptology ePrint Archive, Report 2017/634. 2017 (cit. on p. 65).

[Bou+23] Daniel Bourdrez, Dr. Hugo Krawczyk, Kevin Lewi, and Christopher A. Wood. *The OPAQUE Asymmetric PAKE Protocol*. Internet-Draft draft-irtf-cfrg-opaque-10. Internet Engineering Task Force, Mar. 2023. 70 pp. URL: https://datatracker.ietf.org/doc/draft-irtf-cfrg-opaque/10/ (cit. on p. 175).

[BMS20] Colin Boyd, Anish Mathuria, and Douglas Stebila. *Protocols for Authentication and Key Establishment, Second Edition*. Information Security and Cryptography. Springer, 2020. DOI: 10.1007/978-3-662-58146-9 (cit. on p. 156).

[Boy+05] Michel Boyer, Gilles Brassard, Peter Hoyer, and Alain Tappa. "Tight Bounds on Quantum Searching". In: vol. 46. Jan. 2005, pp. 187–199. ISBN: 9783527603091. DOI: 10.1002/3527603093.ch10 (cit. on p. 34).

[Bra83] G. Brassard. "Relativized cryptography". In: *IEEE Transactions on Information Theory* 29.6 (1983), pp. 877–894. DOI: 10.1109/TIT.1983.1056754 (cit. on p. 7).

[Bra+02] Gilles Brassard, Peter Høyer, Michele Mosca, and Alain Tapp. *Quantum amplitude amplification and estimation*. 2002. DOI: 10.1090/conm/305/05215. URL: http://dx.doi.org/10.1090/conm/305/05215 (cit. on p. 34).

[BK05] Sergey Bravyi and Alexei Kitaev. "Universal quantum computation with ideal Clifford gates and noisy ancillas". In: *Phys. Rev. A* 71 (2 Feb. 2005), p. 022316. DOI: 10.1103/PhysRevA.71.022316 (cit. on pp. 43, 44).

[Brz+11] Christina Brzuska, Marc Fischlin, Bogdan Warinschi, and Stephen C. Williams. "Composability of bellare-rogaway key exchange protocols". In: *Proceedings of the 18th ACM Conference on Computer and Communications Security, CCS 2011, Chicago, Illinois, USA, October 17-21, 2011*. Ed. by Yan Chen, George Danezis, and Vitaly Shmatikov. ACM, 2011, pp. 51–62. DOI: 10.1145/2046707.2046716. URL: https://doi.org/10.1145/2046707.2046716 (cit. on pp. 143, 148, 156, 157, 160).

[BDH11] Johannes Buchmann, Erik Dahmen, and Andreas Hülsing. "XMSS - A Practical Forward Secure Signature Scheme based on Minimal Security Assumptions". In: *IACR Cryptol. ePrint Arch.* (2011), p. 484. URL: http://eprint.iacr.org/2011/484 (cit. on p. 51).

[CKM19] Earl Campbell, Ankur Khurana, and Ashley Montanaro. "Applying quantum algorithms to constraint satisfaction problems". In: *Quantum* 3 (July 2019), p. 167. DOI: 10.22331/q-2019-07-18-167. URL: https://doi.org/10.22331%2Fq-2019-07-18-167 (cit. on p. 112).

[CK02] Ran Canetti and Hugo Krawczyk. "Security Analysis of IKE's Signature-Based Key-Exchange Protocol". In: *Advances in Cryptology - CRYPTO 2002, 22nd Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 2002, Proceedings*. Ed. by Moti Yung. Vol. 2442. Lecture Notes in Computer Science. Springer, 2002, pp. 143–161. DOI: 10.1007/3-540-45708-9_10. URL: https://doi.org/10.1007/3-540-45708-9%5C_10 (cit. on pp. 156, 157, 159, 165–168).

[Cel+22] Sofía Celi, Jonathan Hoyland, Douglas Stebila, and Thom Wiggers. "A Tale of Two Models: Formal Verification of KEMTLS via Tamarin". In: *Computer Security – ESORICS 2022*. Ed. by Vijayalakshmi Atluri, Roberto Di Pietro, Christian D. Jensen, and Weizhi Meng. Cham: Springer Nature Switzerland, 2022, pp. 63–83. ISBN: 978-3-031-17143-7. DOI: 10.1007/978-3-031-17143-7_4 (cit. on pp. 171, 173).

[CL21] André Chailloux and Johanna Loyer. *Lattice sieving via quantum random walks*. 2021. arXiv: 2105.05608 [quant-ph] (cit. on p. 105).

[CNS17] André Chailloux, María Naya-Plasencia, and André Schrottenloher. "An Efficient Quantum Collision Search Algorithm and Implications on Symmetric Cryptography". In: *Advances in Cryptology – ASIACRYPT 2017*. Ed. by Tsuyoshi Takagi and Thomas Peyrin. Cham: Springer International Publishing, 2017, pp. 211–240. ISBN: 978-3-319-70697-9 (cit. on pp. 59, 83, 84, 102, 103).

[Che13] Yuanmi Chen. "Reduction de reseau et securite concrete du chiffrement completement homomorphe". PhD thesis. Paris 7, 2013. URL: https://archive.org/details/PhDChen13 (cit. on p. 125).

[CN11] Yuanmi Chen and Phong Q. Nguyen. "BKZ 2.0: Better Lattice Security Estimates". In: *Advances in Cryptology - ASIACRYPT 2011 - 17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul, South Korea, December 4-8, 2011. Proceedings*. Ed. by Dong Hoon Lee and Xiaoyun Wang. Vol. 7073. Lecture Notes in Computer Science. Springer, 2011, pp. 1–20. DOI: 10.1007/978-3-642-25385-0_1. URL: https://doi.org/10.1007/978-3-642-25385-0_1 (cit. on pp. 61, 105).

[Cle+98] R. Cleve, A. Ekert, C. Macchiavello, and M. Mosca. "Quantum algorithms revisited". In: *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences* 454.1969 (Jan. 1998), pp. 339–354. ISSN: 1471-2946. DOI: 10.1098/rspa.1998.0164. URL: http://dx.doi.org/10.1098/rspa.1998.0164 (cit. on p. 35).

[CG20] Jean-Sébastien Coron and Agnese Gini. "Improved cryptanalysis of the AJPS Mersenne based cryptosystem". In: *Journal of Mathematical Cryptology* 14 (July 2020). DOI: 10.1515/jmc-2019-0027 (cit. on p. 51).

[Cre11] Cas Cremers. "Key Exchange in IPsec Revisited: Formal Analysis of IKEv1 and IKEv2". In: *Computer Security - ESORICS 2011 - 16th European Symposium on Research in Computer Security, Leuven, Belgium, September 12-14, 2011. Proceedings*. Ed. by Vijay Atluri and Claudia Díaz. Vol. 6879. Lecture Notes in Computer Science. Springer, 2011, pp. 315–334. DOI: 10.1007/978-3-642-23822-2_18. URL: https://doi.org/10.1007/978-3-642-23822-2%5C_18 (cit. on p. 157).

[Cry19] Crypto Forum Research Group. *CFRG PAKE Standardization Process*. 2019. URL: https://github.com/cfrg/pake-selection (cit. on p. 140).

[DAn+19a] Jan-Pieter D'Anvers, Qian Guo, Thomas Johansson, Alexander Nilsson, Frederik Vercauteren, and Ingrid Verbauwhede. "Decryption Failure Attacks on IND-CCA Secure Lattice-Based Schemes". In: 2019, pp. 565–598. DOI: 10.1007/978-3-030-17259-6_19 (cit. on pp. 65, 81).

[DRV19] Jan-Pieter D'Anvers, Mélissa Rossi, and Fernando Virdia. *(One) failure is not an option: Bootstrapping the search for failures in lattice-based encryption schemes*. Cryptology ePrint Archive, Report 2019/1399. https://eprint.iacr.org/2019/1399. 2019 (cit. on p. 81).

[DRV20] Jan-Pieter D'Anvers, Mélissa Rossi, and Fernando Virdia. "(One) Failure Is Not an Option: Bootstrapping the Search for Failures in Lattice-Based Encryption Schemes". In: 2020, pp. 3–33. DOI: 10.1007/978-3-030-45727-3_1 (cit. on p. 65).

[DAn+19b] Jan-Pieter D'Anvers, Marcel Tiepelt, Frederik Vercauteren, and Ingrid Verbauwhede. "Timing Attacks on Error Correcting Codes in Post-Quantum Schemes". In: *Proceedings of ACM Workshop on Theory of Implementation Security, TIS at CCS 2019, London, UK, November 11, 2019*. Ed. by Begül Bilgin, Svetla Petkova-Nikova, and Vincent Rijmen. ACM, 2019, pp. 2–9. DOI: 10.1145/3338467.3358948. URL: https://doi.org/10.1145/3338467.3358948 (cit. on pp. 15, 51).

[DVV18] Jan-Pieter D'Anvers, Frederik Vercauteren, and Ingrid Verbauwhede. *On the impact of decryption failures on the security of LWE/LWR based schemes*. Cryptology ePrint Archive, Report 2018/1089. https://eprint.iacr.org/2018/1089. 2018 (cit. on p. 65).

[DVV19] Jan-Pieter D'Anvers, Frederik Vercauteren, and Ingrid Verbauwhede. "The Impact of Error Dependencies on Ring/Mod-LWE/LWR Based Schemes". In: 2019, pp. 103–115. DOI: 10.1007/978-3-030-25510-7_6 (cit. on p. 65).

[DS10] Özgür Dagdelen and Michael Schneider. *Parallel Enumeration of Shortest Lattice Vectors*. Cryptology ePrint Archive, Report 2010/097. https://eprint.iacr.org/2010/097. 2010 (cit. on p. 113).

[De 23] Ronald De Wolf. *Quantum Computing: Lecture Notes*. 2023. URL: https://homepages.cwi.nl/~rdewolf/qcnotes.pdf (cit. on p. 31).

[Det+10] Jérémie Detrey, Guillaume Hanrot, Xavier Pujol, and Damien Stehlé. "Accelerating Lattice Reduction with FPGAs". In: 2010, pp. 124–143 (cit. on p. 121).

[DY83] D. Dolev and A. Yao. "On the Security of Public Key Protocols". In: *IEEE Transactions on Information Theory* 29.2 (1983), pp. 198–208. DOI: 10.1109/TIT.1983.1056650 (cit. on p. 170).

[DM17] Jason A Donenfeld and Kevin Milner. "Formal verification of the WireGuard protocol". In: *Technical Report, Tech. Rep.* (2017). URL: https://www.wireguard.com/papers/wireguard-formal-verification.pdf (cit. on p. 172).

[Dra+06] Thomas G. Draper, Samuel A. Kutin, Eric M. Rains, and Krysta M. Svore. "A Logarithmic-Depth Quantum Carry-Lookahead Adder". In: *Quantum Info. Comput.* 6.4 (July 2006), pp. 351–369. ISSN: 1533-7146 (cit. on pp. 121, 122).

[ES21] Edward Eaton and Douglas Stebila. "The "Quantum Annoying" Property of Password-Authenticated Key Exchange Protocols". In: *Post-Quantum Cryptography - 12th International Workshop, PQCrypto 2021, Daejeon, South Korea, July 20-22, 2021, Proceedings*. Ed. by Jung Hee Cheon and Jean-Pierre Tillich. Vol. 12841. Lecture Notes in Computer Science. Springer, 2021, pp. 154–173. DOI: 10.1007/978-3-030-81293-5_9. URL: https://doi.org/10.1007/978-3-030-81293-5%5C_9 (cit. on pp. 151–153, 175, 176, 179, 180, 185).

[EUR23] EUROCONTROL. *SatCOM*. 2023. URL: https://www.eurocontrol.int/system/satellite-communications-datalink (cit. on p. 156).

[FP85] Ulrich Fincke and Michael Pohst. "Improved methods for calculating vectors of short length in a lattice, including a complexity analysis". In: *Mathematics of computation* 44.170 (1985), pp. 463–471 (cit. on pp. 61, 62, 105).

[FDJ13] Austin G. Fowler, Simon J. Devitt, and Cody Jones. "Surface code implementation of block code state distillation". In: *Scientific Reports* 3.1 (June 2013), p. 1939. ISSN: 2045-2322. DOI: 10.1038/srep01939 (cit. on pp. 42, 44, 87, 100, 101).

[Fow+12] Austin G. Fowler, Matteo Mariantoni, John M. Martinis, and Andrew N. Cleland. "Surface codes: Towards practical large-scale quantum computation". In: *Phys. Rev. A* 86 (3 Sept. 2012), p. 032324. DOI: 10.1103/PhysRevA.86.032324 (cit. on p. 41).

[FO99] Eiichiro Fujisaki and Tatsuaki Okamoto. "Secure Integration of Asymmetric and Symmetric Encryption Schemes". In: *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*. Ed. by Michael J. Wiener. Vol. 1666. Lecture Notes in Computer Science. Springer, 1999, pp. 537–554. DOI: 10.1007/3-540-48405-1_34. URL: https://doi.org/10.1007/3-540-48405-1%5C_34 (cit. on p. 48).

[FO13] Eiichiro Fujisaki and Tatsuaki Okamoto. "Secure Integration of Asymmetric and Symmetric Encryption Schemes". In: *Journal of Cryptology*. 2013. DOI: 10.1007/s00145-011-9114-1 (cit. on p. 48).

[GN08a] Nicolas Gama and Phong Q. Nguyen. "Finding short lattice vectors within Mordell's inequality". In: 2008, pp. 207–216. DOI: 10.1145/1374376.1374408 (cit. on p. 61).

[GN08b] Nicolas Gama and Phong Q. Nguyen. "Predicting Lattice Reduction". In: *Advances in Cryptology – EUROCRYPT 2008*. Ed. by Nigel Smart. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 31–51. ISBN: 978-3-540-78967-3 (cit. on p. 50).

[GNR10] Nicolas Gama, Phong Q. Nguyen, and Oded Regev. "Lattice Enumeration Using Extreme Pruning". In: 2010, pp. 257–278. DOI: 10.1007/978-3-642-13190-5_13 (cit. on pp. 61, 62, 64, 105, 113–115, 118, 119, 127, 132).

[Gaz+21] Stefan-Lukas Gazdag, Sophia Grundner-Culemann, Tobias Guggemos, Tobias Heider, and Daniel Loebenberger. "A Formal Analysis of IKEv2's Post-Quantum Extension". In: *Proceedings of the 37th Annual Computer Security Applications Conference*. ACSAC '21. New York, NY, USA: Association for Computing Machinery, 2021, pp. 91–105. ISBN: 9781450385794. DOI: 10.1145/3485832.3485885. URL: https://doi.org/10.1145/3485832.3485885 (cit. on pp. 157, 171, 172).

[GM19] Vlad Gheorghiu and Michele Mosca. *Benchmarking the quantum cryptanalysis of symmetric, public-key and hash-based cryptographic schemes*. arXiv:1902.02332. 2019. arXiv: 1902.02332 [quant-ph] (cit. on p. 149).

[GE21] Craig Gidney and Martin Ekerå. "How to factor 2048 bit RSA integers in 8 hours using 20 million noisy qubits". In: *Quantum* 5 (2021), p. 433. DOI: 10.22331/q-2021-04-15-433. URL: https://doi.org/10.22331/q-2021-04-15-433 (cit. on pp. 7, 44, 139, 149).

[GLM08] Vittorio Giovannetti, Seth Lloyd, and Lorenzo Maccone. "Quantum Random Access Memory". In: *Phys. Rev. Lett.* 100 (16 Apr. 2008), p. 160501. DOI: 10.1103/PhysRevLett.100.160501. URL: https://link.aps.org/doi/10.1103/PhysRevLett.100.160501 (cit. on pp. 33, 106, 115).

[GJK21] Yanqi Gu, Stanislaw Jarecki, and Hugo Krawczyk. "KHAPE: Asymmetric PAKE from Key-Hiding Key Exchange". In: *Advances in Cryptology - CRYPTO 2021 - 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16-20, 2021, Proceedings, Part IV*. Ed. by Tal Malkin and Chris Peikert. Vol. 12828. Lecture Notes in Computer Science. Springer, 2021, pp. 701–730. DOI: 10.1007/978-3-030-84259-8_24. URL: https://doi.org/10.1007/978-3-030-84259-8%5C_24 (cit. on pp. 176–178).

[GJS16] Qian Guo, Thomas Johansson, and Paul Stankovski. *A Key Recovery Attack on MDPC with CCA Security Using Decoding Errors*. Cryptology ePrint Archive, Report 2016/858. https://eprint.iacr.org/2016/858. 2016 (cit. on p. 65).

[GJY19] Qian Guo, Thomas Johansson, and Jing Yang. "A Novel CCA Attack Using Decryption Errors Against LAC". In: 2019, pp. 82–111. DOI: 10.1007/978-3-030-34578-5_4 (cit. on p. 65).

[Hän+20] Thomas Häner, Samuel Jaques, Michael Naehrig, Martin Roetteler, and Mathias Soeken. "Improved Quantum Circuits for Elliptic Curve Discrete Logarithms". In: *Post-Quantum Cryptography*. Ed. by Jintai Ding and Jean-Pierre Tillich. Cham: Springer International Publishing, 2020, pp. 425–444. ISBN: 978-3-030-44223-1 (cit. on pp. 121, 122).

[HRS17] Thomas Häner, Martin Roetteler, and Krysta M. Svore. "Factoring Using 2n + 2 Qubits with Toffoli Based Modular Multiplication". In: 17.7-8 (June 2017), pp. 673–684. ISSN: 1533-7146 (cit. on p. 121).

[HO22] Feng Hao and Paul C. van Oorschot. "SoK: Password-Authenticated Key Exchange – Theory, Practice, Standardization and Real-World Lessons". In: ASIA CCS '22. Nagasaki, Japan: Association for Computing Machinery, 2022, pp. 697–711. ISBN: 9781450391405. DOI: 10.1145/3488932.3523256. URL: https://doi.org/10.1145/3488932.3523256 (cit. on p. 175).

[Her+10] Jens Hermans, Michael Schneider, Johannes Buchmann, Frederik Vercauteren, and Bart Preneel. "Parallel Shortest Lattice Vector Enumeration on Graphics Cards". In: 2010, pp. 52–68 (cit. on pp. 113, 120, 121).

[HYY23] Minki Hhan, Takashi Yamakawa, and Aaram Yun. *Quantum Complexity for Discrete Logarithms and Related Problems*. Cryptology ePrint Archive, Paper 2023/1054. https://eprint.iacr.org/2023/1054. 2023. URL: https://eprint.iacr.org/2023/1054 (cit. on p. 177).

[HYY24] Minki Hhan, Takashi Yamakawa, and Aaram Yun. "Quantum Complexity for Discrete Logarithms and Related Problems". In: *Advances in Cryptology – CRYPTO 2024*. Ed. by Leonid Reyzin and Douglas Stebila. Cham: Springer Nature Switzerland, 2024, pp. 3–36. ISBN: 978-3-031-68391-6 (cit. on p. 177).

[HKT10] Thomas Holenstein, Robin Künzler, and Stefano Tessaro. "Equivalence of the Random Oracle Model and the Ideal Cipher Model, Revisited". In: *Computing Research Repository - CORR* (Nov. 2010). DOI: 10.1145/1993636.1993650 (cit. on p. 22).

[How+03] Nick Howgrave-Graham, Phong Q. Nguyen, David Pointcheval, John Proos, Joseph H. Silverman, Ari Singer, and William Whyte. "The Impact of Decryption Failures on the Security of NTRU Encryption". In: *Advances in Cryptology - CRYPTO 2003*. Ed. by Dan Boneh. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 226–246. ISBN: 978-3-540-45146-4 (cit. on p. 65).

[Hül+20] Andreas Hülsing, Daniel J. Bernstein, Christoph Dobraunig, Maria Eichlseder, Scott Fluhrer, Stefan-Lukas Gazdag, Pannos Kampanakis, Stefan Kolbl, Tanja Lange, Martin M Lauridsen, Florian Mendel, Ruben Niederhagen, Christian Rechberger, Joost Rijneveld, Peter Schwabe, Jean-Philippe Aumasson, Bas Westerbaan, and Ward Beullens. *SPHINCS+-Submission to the 3rd round of the NIST post-quantum project*. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/post-quantum-cryptography-standardization/post-quantum-cryptography-standardization/round-3-submissions. 2020 (cit. on pp. 11, 29, 52–54, 56, 58, 89, 90, 93–95).

[HRS16] Andreas Hülsing, Joost Rijneveld, and Fang Song. "Mitigating Multi-target Attacks in Hash-Based Signatures". In: 2016, pp. 387–416. DOI: 10.1007/978-3-662-49384-7_15 (cit. on pp. 23, 52, 58).

[IBM19] IBM. *On quantum supremacy*. 2019. URL: https://www.ibm.com/quantum/blog/on-quantum-supremacy (cit. on p. 6).

[IEC17] IEC. *Information technology – Personal identification – ISO-compliant driving licence*. ISO/IEC 18013-3:2027. 2017 (cit. on p. 175).

[IEE09] IEEE. *IEEE Standard Specification for Password-Based Public-Key Cryptographic Techniques*. IEEE Std 1363.2-2008. 2009. DOI: 10.1109/IEEESTD.2009.4773330 (cit. on p. 175).

[Int23] International Civil Aviation organization. *CHAPTER 13 L-Band Digital Aeronautical Communications System (LDACS)*. Tech. rep. International Civil Aviation organization (ICAO), 2023, pp. 1–15. URL: https://www.ldacs.com/wp-content/uploads/2023/03/WP06.AppA-DCIWG-6-LDACS_SARPs.pdf (cit. on pp. 160, 162).

[Int21] International Civil Aviation Organization (ICAO). *ICAO - ANNEX 10 VOL III AMD 91 Aeronautical Telecommunications Volume III - Communications Systems (Part I - Digital Data Communication Systems; Part II - Voice Communication Systems)*. Mar. 2021 (cit. on pp. 140, 155, 160).

[ISO21] ISO copyright office. *ISO/IEC 11779-3*. 2021. URL: https://www.iso.org/standard/82709.html (cit. on pp. 155, 156, 160, 172).

[Jaq+20] Samuel Jaques, Michael Naehrig, Martin Roetteler, and Fernando Virdia. "Implementing Grover Oracles for Quantum Key Search on AES and LowMC". In: 2020, pp. 280–310. DOI: 10.1007/978-3-030-45724-2_10 (cit. on pp. 6, 40, 41, 44, 45, 123, 127, 129, 131, 134, 136).

[JR23] Samuel Jaques and Arthur G. Rattew. *QRAM: A Survey and Critique*. 2023. arXiv: 2305.10310 [quant-ph] (cit. on pp. 33, 106, 115).

[JS19] Samuel Jaques and John M. Schanck. “Quantum Cryptanalysis in the RAM Model: Claw-Finding Attacks on SIKE”. In: 2019, pp. 32–61. DOI: 10.1007/978-3-030-26948-7_2 (cit. on pp. 40, 126).

[Jaq19] Jaques, Samuel. “Quantum Cost Models for Cryptanalysis of Isogenies”. MA thesis. University of Waterloo, 2019. URL: http://hdl.handle.net/10012/14612 (cit. on pp. 42, 43).

[JKX18] Stanislaw Jarecki, Hugo Krawczyk, and Jiayu Xu. “OPAQUE: An Asymmetric PAKE Protocol Secure Against Pre-computation Attacks”. In: *Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part III*. Ed. by Jesper Buus Nielsen and Vincent Rijmen. Vol. 10822. Lecture Notes in Computer Science. Springer, 2018, pp. 456–486. DOI: 10.1007/978-3-319-78372-7_15. URL: https://doi.org/10.1007/978-3-319-78372-7%5C_15 (cit. on p. 176).

[JJ00] Éliane Jaulmes and Antoine Joux. “A Chosen-Ciphertext Attack against NTRU”. In: 2000, pp. 20–35. DOI: 10.1007/3-540-44598-6_2 (cit. on p. 65).

[Jon+12] N. Cody Jones, Rodney Van Meter, Austin G. Fowler, Peter L. McMahon, Jungsang Kim, Thaddeus D. Ladd, and Yoshihisa Yamamoto. “Layered Architecture for Quantum Computing”. In: *Phys. Rev. X* 2 (3 July 2012), p. 031007. DOI: 10.1103/PhysRevX.2.031007 (cit. on pp. 38, 39, 41–43).

[Jon+10] Nathan Jones, Rodney Van Meter, Austin Fowler, Peter McMahon, Jungsang Kim, Thaddeus Ladd, and Yoshihisa Yamamoto. “Layered Architecture for Quantum Computing”. In: *Physical Review X* 2 (Oct. 2010). DOI: 10.1103/PhysRevX.2.031007 (cit. on pp. 40, 41).

[Kan83] Ravi Kannan. “Improved Algorithms for Integer Programming and Related Lattice Problems”. In: 1983, pp. 193–206. DOI: 10.1145/800061.808749 (cit. on pp. 61, 62, 105).

[Kan87] Ravi Kannan. “Minkowski’s Convex Body Theorem and Integer Programming”. In: *Math. Oper. Res.* 12 (1987), pp. 415–440 (cit. on p. 60).

[Kau+14a] C. Kaufman, P. Hoffman, Y. Nir, P. Eronen, and T. Kivinen. *Internet Key Exchange Protocol Version 2*. https://datatracker.ietf.org/doc/html/rfc7296, accessed Oct 01, 2023. Additional authors: Microsoft, VPN Consortium, Check Point, Independent, INSIDE SECURE. 2014. DOI: 10.17487/RFC7296 (cit. on pp. 156, 163).

[Kau+14b] C. Kaufman, P. Hoffman, Y. Nir, P. Eronen, and T. Kivinen. *Minimal Internet Key Exchange Protocol Version 2*. https://www.rfc-editor.org/rfc/rfc7815, accessed Oct 01, 2023. Additional authors: Microsoft, VPN Consortium, Check Point, Independent, INSIDE SECURE. 2014. DOI: 10.17487/RFC7815 (cit. on p. 163).

[Ken99] Adrian Kent. “Unconditionally Secure Bit Commitment”. In: *Physical Review Letters* 83.7 (Aug. 1999), pp. 1447–1450. ISSN: 1079-7114. DOI: 10.1103/physrevlett.83.1447. URL: http://dx.doi.org/10.1103/PhysRevLett.83.1447 (cit. on p. 4).

[Kir+19] Elena Kirshanova, Erik Mårtensson, Eamonn W. Postlethwaite, and Subhayan Roy Moulik. “Quantum Algorithms for the Approximate k-List Problem and Their Application to Lattice Sieving”. In: 2019, pp. 521–551. DOI: 10.1007/978-3-030-34578-5_19 (cit. on p. 105).

[Koc+22] Daniel Koch, Michael Samodurov, Andrew Projansky, and Paul M Alsing. “Gate-based circuit designs for quantum adder-inspired quantum random walks on superconducting qubits”. In: *International Journal of Quantum Information* 20.03 (2022), p. 2150043 (cit. on p. 121).

[Köl+16] Stefan Kölbl, Martin M. Lauridsen, Florian Mendel, and Christian Rechberger. “Haraka v2 - Efficient Short-Input Hashing for Post-Quantum Applications”. In: 2016.2 (2016). https://tosc.iacr.org/index.php/ToSC/article/view/563, pp. 1–29. DOI: 10.13154/tosc.v2016.i2.1-29 (cit. on pp. 29, 53, 59).

[Kra03] Hugo Krawczyk. “SIGMA: The ’SIGn-and-MAc’ Approach to Authenticated Diffie-Hellman and Its Use in the IKE-Protocols”. In: *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings*. Ed. by Dan Boneh. Vol. 2729. Lecture Notes in Computer Science. Springer, 2003, pp. 400–425. DOI: 10.1007/978-3-540-45146-4_24. URL: https://doi.org/10.1007/978-3-540-45146-4%5C_24 (cit. on pp. 156, 159).

[Kra05] Hugo Krawczyk. “HMQV: A High-Performance Secure Diffie-Hellman Protocol”. In: *Advances in Cryptology - CRYPTO 2005: 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings*. Ed. by Victor Shoup. Vol. 3621. Lecture Notes in Computer Science. Springer, 2005, pp. 546–566. DOI: 10.1007/11535218_33. URL: https://doi.org/10.1007/11535218%5C_33 (cit. on p. 177).

[Kuo+11] Po-Chun Kuo, Michael Schneider, Özgür Dagdelen, Jan Reichelt, Johannes Buchmann, Chen-Mou Cheng, and Bo-Yin Yang. “Extreme Enumeration on GPU and in Clouds - - How Many Dollars You Need to Break SVP Challenges -”. In: 2011, pp. 176–191. DOI: 10.1007/978-3-642-23951-9_12 (cit. on p. 113).

[Kup11] Greg Kuperberg. *Another subexponential-time quantum algorithm for the dihedral hidden subgroup problem*. arXiv preprint arXiv:1112.3333. 2011. arXiv: 1112.3333 [quant-ph] (cit. on pp. 33, 106, 115).

[Laa15] Thijs Laarhoven. "Sieving for Shortest Vectors in Lattices Using Angular Locality-Sensitive Hashing". In: 2015, pp. 3–22. DOI: 10.1007/978-3-662-47989-6_1 (cit. on pp. 61, 105).

[LMv13] Thijs Laarhoven, Michele Mosca, and Joop van de Pol. "Solving the Shortest Vector Problem in Lattices Faster Using Quantum Search". In: 2013, pp. 83–101. DOI: 10.1007/978-3-642-38616-9_6 (cit. on pp. 12, 105).

[Lam79] Leslie Lamport. "Constructing Digital Signatures from a One Way Function". In: 1979 (cit. on p. 51).

[LN20] Jianwei Li and Phong Q. Nguyen. *A Complete Analysis of the BKZ Lattice Reduction Algorithm*. Cryptology ePrint Archive, Report 2020/1237. https://eprint.iacr.org/2020/1237. 2020 (cit. on pp. 61, 105).

[LP11] Richard Lindner and Chris Peikert. "Better Key Sizes (and Attacks) for LWE-Based Encryption". In: 2011, pp. 319–339. DOI: 10.1007/978-3-642-19074-2_21 (cit. on pp. 61, 105).

[LC96] Hoi-Kwong Lo and H. F. Chau. *Why quantum bit commitment and ideal quantum coin tossing are impossible*. 1996. arXiv: quant-ph/9605026 (cit. on p. 4).

[Low97] G. Lowe. "A Hierarchy of Authentication Specifications". In: *Proceedings 10th Computer Security Foundations Workshop*. Rockport, MA, USA, 1997, pp. 31–43. DOI: 10.1109/CSFW.1997.596782 (cit. on p. 171).

[Lyu+22] Vadim Lyubashevsky, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Peter Schwabe, Gregor Seiler, Damien Stehlé, and Shi Bai. *CRYSTALS-DILITHIUM*. Tech. rep. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022. National Institute of Standards and Technology, 2022 (cit. on pp. 30, 61, 105).

[LM09] Vadim Lyubashevsky and Daniele Micciancio. "On Bounded Distance Decoding, Unique Shortest Vectors, and the Minimum Distance Problem". In: 2009, pp. 577–594. DOI: 10.1007/978-3-642-03356-8_34 (cit. on p. 60).

[Mäu+21] Nils Mäurer, Thomas Gräupl, Christoph Gentsch, Tobias Guggemos, Marcel Tiepelt, Corinna Schmitt, and Gabi Dreo Rodosek. "A Secure Cell-Attachment Procedure of LDACS". In: *IEEE European Symposium on Security and Privacy Workshops, EuroS&P 2021, Vienna, Austria, September 6-10, 2021*. IEEE, 2021, pp. 113–122. DOI: 10.1109/EuroSPW54576.2021.00019. URL: https://doi.org/10.1109/EuroSPW54576.2021.00019 (cit. on pp. 16, 157).

[MGS21] Nils Mäurer, Thomas Gräupl, and Corinna Schmitt. *Cybersecurity for the L-band Digital Aeronautical Communications System (LDACS)*. Tech. rep. German Aerospace Center, June 2021, pp. 1–. DOI: 10.1049/SBRA545E_ch4 (cit. on p. 157).

[MG22] Nils Mäurer and Sophia Grundner-Culemann. *Formal Verification of the LDACS MAKE Protocol*. crypto day matters 34. Bonn, 2022. DOI: 10.18420/cdm-2022-34-24 (cit. on p. 157).

[May97] Dominic Mayers. "Unconditionally Secure Quantum Bit Commitment is Impossible". In: *Physical Review Letters* 78.17 (Apr. 1997), pp. 3414–3417. ISSN: 1079-7114. DOI: 10.1103/physrevlett.78.3414. URL: http://dx.doi.org/10.1103/PhysRevLett.78.3414 (cit. on p. 4).

[Mei+13] S. Meier, B. Schmidt, C. Cremers, and D. Basin. "The TAMARIN Prover For The Symbolic Analysis Of Security Protocols". In: *25th International Conference on Computer Aided Verification (CAV)*. Saint Petersburg, Russia, 2013, pp. 696–701. DOI: 10.1007/978-3-642-39799-8_48 (cit. on p. 170).

[Mel+19] Carlos Aguilar Melchor, Nicolas Aragon, Magali Bardet, Slim Bettaieb, Loic Bidoux, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, Adrien Hauteville, Ayoub Otmani, Olivier Ruatta, Jean-Piere Tillich, and Gilles Zemor. *ROLLO-Rank-Ouroboros, LAKE & LOCKER*. Technical report, National Institute of Standards and Technology. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-2-submissions. 2019 (cit. on p. 65).

[Mel+18] Carlos Aguilar Melchor, Nicolas Aragon, Slim Bettaieb, Loïc Bidoux, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, Edoardo Persichetti, and GillesZémor. *Hamming Quasi-Cyclic (HQC)*. Technical report, National Institute of Standards and Technology. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-2-submissions. 2018 (cit. on p. 65).

[MW15] Daniele Micciancio and Michael Walter. "Fast Lattice Point Enumeration with Minimal Overhead". In: 2015, pp. 276–294. DOI: 10.1137/1.9781611973730.21 (cit. on p. 62).

[MW16] Daniele Micciancio and Michael Walter. "Practical, Predictable Lattice Basis Reduction". In: 2016, pp. 820–849. DOI: 10.1007/978-3-662-49890-3_31 (cit. on p. 61).

[Mic20] Microsoft. *Q# Language Specification*. 2020. URL: https://github.com/microsoft/qsharp-language/tree/main/Specifications/Language#q-language (cit. on p. 40).

[Mon18] Ashley Montanaro. “Quantum-Walk Speedup of Backtracking Algorithms”. In: *Theory Comput.* 14.1 (2018), pp. 1–24. DOI: 10.4086/toc.2018.v014a015. URL: https://doi.org/10.4086/toc.2018.v014a015 (cit. on pp. 12, 35–37, 105, 107–112, 120, 121).

[MT19] Edgard Munoz-Coreas and Himanshu Thapliyal. “Quantum Circuit Design of a T-count Optimized Integer Multiplier”. In: *IEEE Transactions on Computers* 68.5 (2019), pp. 729–739. DOI: 10.1109/TC.2018.2882774 (cit. on p. 121).

[Nae+17] Michael Naehrig, Erdem Alkim, Joppe Bos, Leo 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. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-1-submissions. 2017 (cit. on p. 65).

[Nat15] National Institute for Standards and Technology. *Secure Hash Standard*. FIPS 180-4. 2015. URL: https://csrc.nist.gov/pubs/fips/180-4/upd1/final (cit. on p. 161).

[Nat17] National Institute for Standards and Technology. *Post-Quantum Cryptography Call for Proposals*. 2017. URL: https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/call-for-proposals-final-dec-2016.pdf (cit. on pp. 7, 9, 27, 40, 44, 78, 106, 124, 139).

[Nat20] National Institute for Standards and Technology. *Post-Quantum Cryptography Round 1*. 2020. URL: https://csrc.nist.gov/Projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-1-submissions (cit. on pp. 52, 65, 66).

[Nat22] National Institute for Standards and Technology. *NIST: Selected Algorithms 2022*. 2022. URL: https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022 (cit. on pp. 11, 12, 83, 155, 157, 175).

[Nat24a] National Institute for Standards and Technology. *FIPS 203*. 2024. URL: https://csrc.nist.gov/pubs/fips/203/ipd (cit. on p. 12).

[Nat24b] National Institute for Standards and Technology. *FIPS 204*. 2024. URL: https://csrc.nist.gov/pubs/fips/204/final (cit. on p. 12).

[Nat24c] National Institute for Standards and Technology. *FIPS 205*. 2024. URL: https://csrc.nist.gov/pubs/fips/205/ipd (cit. on p. 11).

[NS78] Roger M. Needham and Michael D. Schroeder. “Using Encryption for Authentication in Large Networks of Computers”. In: *Commun. ACM* 21.12 (Dec. 1978), pp. 993–999. ISSN: 0001-0782. DOI: 10.1145/359657.359659. URL: https://doi.org/10.1145/359657.359659 (cit. on p. 170).

[NV08] Phong Q. Nguyen and Thomas Vidick. “Sieve algorithms for the shortest vector problem are practical”. In: *J. Math. Cryptol.* 2.2 (2008), pp. 181–207. DOI: 10.1515/JMC.2008.009. URL: https://doi.org/10.1515/JMC.2008.009 (cit. on pp. 61, 105).

[Nie+23] Junhong Nie, Qinlin Zhu, Meng Li, and Xiaoming Sun. “Quantum circuit design for integer multiplication based on Schönhage-Strassen algorithm”. In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* (2023), pp. 1–1. DOI: 10.1109/TCAD.2023.3279300 (cit. on pp. 121, 122).

[NC11] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2011. ISBN: 9781107002173 (cit. on p. 31).

[NIS15] NIST. *SHA-3 standard: Permutation-based hash and extendable-output functions*. National Institute for Standards and Technology. 2015. DOI: 10.6028/NIST.FIPS.202 (cit. on pp. 53, 58).

[PF14] Monica Paolini and Senza Fili. *Enabling the Next Generation in Air Traffic Management with AeroMACS*. https://files.wimaxforum.org/Document/Download/AeroMACS-Delivering_Next_Generation_Communications_to_the_Airport_Surface. 2014 (cit. on p. 156).

[PV23] Edward Parker and Michael J. D. Vermeer. *Estimating the Energy Requirements to Operate a Cryptanalytically Relevant Quantum Computer*. arXiv:2304.14344. 2023. arXiv: 2304.14344 [quant-ph] (cit. on p. 149).

[Pei14] Chris Peikert. “Lattice Cryptography for the Internet”. In: *Post-Quantum Cryptography - 6th International Workshop, PQCrypto 2014, Waterloo, ON, Canada, October 1-3, 2014. Proceedings*. Ed. by Michele Mosca. Vol. 8772. Lecture Notes in Computer Science. Springer, 2014, pp. 197–219. DOI: 10.1007/978-3-319-11659-4_12. URL: https://doi.org/10.1007/978-3-319-11659-4%5C_12 (cit. on pp. 157, 159, 162, 165).

[Pei16] Chris Peikert. “A Decade of Lattice Cryptography”. In: *Found. Trends Theor. Comput. Sci.* 10.4 (2016), pp. 283–424. DOI: 10.1561/0400000074. URL: https://doi.org/10.1561/0400000074 (cit. on pp. 59, 166).

[PS13] Paul Pham and Krysta M Svore. "A 2D nearest-neighbor quantum architecture for factoring in polylogarithmic depth." In: *Quantum Inf. Comput.* 13.11-12 (2013), pp. 937–962 (cit. on p. 121).

[Poh81] Michael Pohst. "On the Computation of Lattice Vectors of Minimal Length, Successive Minima and Reduced Bases with Applications". In: *SIGSAM Bull.* 15.1 (Feb. 1981), pp. 37–44. ISSN: 0163-5824. DOI: 10.1145/1089242.1089247. URL: https://doi.org/10.1145/1089242.1089247 (cit. on p. 62).

[Pre18] John Preskill. "Quantum Computing in the NISQ era and beyond". In: *Quantum* 2 (Aug. 2018), p. 79. ISSN: 2521-327X. DOI: 10.22331/q-2018-08-06-79. URL: https://doi.org/10.22331/q-2018-08-06-79 (cit. on pp. 64, 106).

[Pre+22] Thomas Prest, Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Ricosset, Gregor Seiler, William Whyte, and Zhenfei Zhang. *FALCON*. Tech. rep. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022. National Institute of Standards and Technology, 2022 (cit. on pp. 30, 61, 105).

[PS08] Xavier Pujol and Damien Stehlé. "Rigorous and Efficient Short Lattice Vectors Enumeration". In: 2008, pp. 390–405. DOI: 10.1007/978-3-540-89255-7_24 (cit. on p. 121).

[Reg05] Oded Regev. "On lattices, learning with errors, random linear codes, and cryptography". In: *Proceedings of the 37th Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, May 22-24, 2005*. Ed. by Harold N. Gabow and Ronald Fagin. ACM, 2005, pp. 84–93. DOI: 10.1145/1060590.1060603. URL: https://doi.org/10.1145/1060590.1060603 (cit. on p. 105).

[Reg09] Oded Regev. "On lattices, learning with errors, random linear codes, and cryptography". In: *J. ACM* 56.6 (2009), 34:1–34:40. DOI: 10.1145/1568318.1568324. URL: https://doi.org/10.1145/1568318.1568324 (cit. on p. 105).

[Flu+20] S. Fluhrer, P. Kampanakis, D. McGrew, and V. Smyslov. *Mixing Preshared Keys in the Internet Key Exchange Protocol Version 2 (IKEv2) for Post-quantum Security*. RFC 8784 (Proposed Standard). RFC. Fremont, CA, USA: RFC Editor, June 2020. DOI: 10.17487/RFC8784. URL: https://www.rfc-editor.org/rfc/rfc8784.txt (cit. on p. 157).

[Roe+17] Martin Roetteler, Michael Naehrig, Krysta M. Svore, and Kristin Lauter. "Quantum Resource Estimates for Computing Elliptic Curve Discrete Logarithms". In: *Advances in Cryptology – ASIACRYPT 2017*. Ed. by Tsuyoshi Takagi and Thomas Peyrin. Cham: Springer International Publishing, 2017, pp. 241–270. ISBN: 978-3-319-70697-9 (cit. on pp. 42, 43, 149).

[RG17] Lidia Ruiz-Perez and Juan Carlos Garcia-Escartin. "Quantum arithmetic with the quantum Fourier transform". In: *Quantum Information Processing* 16 (2017), pp. 1–14 (cit. on p. 121).

[SFW19] Cyprien Delpech de Saint Guilhem, Marc Fischlin, and Bogdan Warinschi. *Authentication in Key-Exchange: Definitions, Relations and Composition*. Cryptology ePrint Archive, Paper 2019/1203. https://eprint.iacr.org/2019/1203. 2019. DOI: 10.1109/CSF49147.2020.00028. URL: https://eprint.iacr.org/2019/1203 (cit. on pp. 143–148, 157–159, 169, 170).

[SFW20] Cyprien Delpech de Saint Guilhem, Marc Fischlin, and Bogdan Warinschi. "Authentication in Key-Exchange: Definitions, Relations and Composition". In: *33rd IEEE Computer Security Foundations Symposium, CSF 2020, Boston, MA, USA, June 22-26, 2020*. IEEE, 2020, pp. 288–303. DOI: 10.1109/CSF49147.2020.00028. URL: https://doi.org/10.1109/CSF49147.2020.00028 (cit. on pp. 143, 160).

[Sch+17] John M. Schanck, Andreas Hulsing, Joost Rijneveld, and Peter Schwabe. *NTRU-HRSS-KEM*. Tech. rep. available at https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-1-submissions. National Institute of Standards and Technology, 2017 (cit. on p. 105).

[Sch17] Joern-Marc Schmidt. *Requirements for Password-Authenticated Key Agreement (PAKE) Schemes*. RFC 8125. Apr. 2017. DOI: 10.17487/RFC8125. URL: https://www.rfc-editor.org/info/rfc8125 (cit. on p. 175).

[SB17] Markus Schmidt and Nina Bindel. *Estimation of the Hardness of the Learning with Errors Problem with a Restricted Number of Samples*. Cryptology ePrint Archive, Report 2017/140. https://eprint.iacr.org/2017/140. 2017 (cit. on p. 60).

[SE94] Claus Schnorr and M. Euchner. "Lattice Basis Reduction: Improved Practical Algorithms and Solving Subset Sum Problems". In: *Mathematical Programming* 66 (Aug. 1994), pp. 181–199. DOI: 10.1007/BF01581144 (cit. on pp. 61, 62, 105, 113).

[Sch20] Mark Schultz-Wu. *Meaning of "Security can be reduced to a problem"*. 2020. URL: %5Curl%7Bhttps://crypto.stackexchange.com/questions/80981/meaning-of-security-can-be-reduced-to-a-problem%7D (cit. on p. 19).

[Sch+22] Peter Schwabe, Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Gregor Seiler, Damien Stehlé, and Jintai Ding. *CRYSTALS-KYBER*. Tech. rep. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022. National Institute of Standards and Technology, 2022 (cit. on pp. 30, 61, 105, 113, 124, 125).

[SES23] SESAR JU. *LDACS A/G Specification, Edition 01.01.00, Template Edition 02.00.05, Edition date 25.04.2023*. Tech. rep. Version PJ.14-W2-60 TRL6 Final LDACS A/G Specification. Earlier version can be found here https://www.ldacs.com/wp-content/uploads/2023/03/SESAR2020_PJ14-W2-60_TRL6_D3_1_230_3rd_LDACS_AG_Specification_v1.0.0.pdf. SESAR JU, 2023. URL: https://www.ldacs.com/publications-and-links/#post-Specifications (cit. on pp. 13, 16, 155–157, 159, 160, 162).

[Sho94] P.W. Shor. "Algorithms for quantum computation: discrete logarithms and factoring". In: *Proceedings 35th Annual Symposium on Foundations of Computer Science*. 1994, pp. 124–134. DOI: 10.1109/SFCS.1994.365700 (cit. on p. 7).

[Sin23] Single European Sky ATM Research Joint Undertaking. *Single European Sky ATM Research*. [Online; accessed 29 December 2023]. 2023. URL: https://www.sesarju.eu/ (cit. on p. 155).

[Sze04] Mario Szegedy. "Quantum Speed-Up of Markov Chain Based Algorithms". In: 2004, pp. 32–41. DOI: 10.1109/FOCS.2004.53 (cit. on p. 35).

[Sze17] Alan Szepieniec. *Ramstake*. Tech. rep. available at https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-1-submissions. National Institute of Standards and Technology, 2017 (cit. on pp. 10, 11, 28, 47–49, 65–68).

[TW22] Tim Taubert and Christopher A. Wood. *SPAKE2+, an Augmented PAKE*. Internet-Draft draft-bar-cfrg-spake2plus-08. Work in Progress. Internet Engineering Task Force, May 2022. 30 pp. URL: https://datatracker.ietf.org/doc/draft-bar-cfrg-spake2plus/08/ (cit. on p. 175).

[tea23] The FPLLL development team. "fplll, a lattice reduction library, Version: 5.4.2". Available at https://github.com/fplll/fplll. 2023. URL: https://github.com/fplll/fplll (cit. on pp. 121–123).

[The23] The Tamarin Team. *Tamarin-Prover Manual*. [Online; accessed 29 December 2023]. 2023. URL: https://tamarin-prover.com/manual/master/tex/tamarin-manual.pdf (cit. on p. 171).

[Tho19] Steve Thomas. *"Re: [Cfrg] Proposed PAKE Selection Process"*. CFRG Mailing List. June 2019. URL: https://mailarchive.ietf.org/arch/msg/cfrg/dtf91cmavpzT47U3AVxrVGNB5UM/# (cit. on p. 140).

[TD20a] Marcel Tiepelt and Jan-Pieter D'Anvers. "Exploiting Decryption Failures in Mersenne Number Cryptosystems". In: *Proceedings of the 7th on ASIA Public-Key Cryptography Workshop, APKC at AsiaCCS 2020, Taipei, Taiwan, October 6, 2020*. Ed. by Keita Emura and Naoto Yanai. ACM, 2020, pp. 45–54. DOI: 10.1145/3384940.3388957. URL: https://doi.org/10.1145/3384940.3388957 (cit. on pp. 11, 17, 28, 31, 47, 65).

[TD20b] Marcel Tiepelt and Jan-Pieter D'Anvers. *Exploiting Decryption Failures in Mersenne Number Cryptosystems*. Cryptology ePrint Archive, Paper 2020/367. https://eprint.iacr.org/2020/367. 2020. DOI: 10.1145/3384940.3388957. URL: https://eprint.iacr.org/2020/367 (cit. on pp. 11, 17, 31, 47, 65).

[TES23a] Marcel Tiepelt, Edward Eaton, and Douglas Stebila. *Making an Asymmetric PAKE Quantum-Annoying by Hiding Group Elements*. Cryptology ePrint Archive, Paper 2023/1513. https://eprint.iacr.org/2023/1513. 2023. DOI: 10.1007/978-3-031-50594-2_9. URL: https://eprint.iacr.org/2023/1513 (cit. on pp. 15, 17, 143, 175).

[TES23b] Marcel Tiepelt, Edward Eaton, and Douglas Stebila. "Making an Asymmetric PAKE Quantum-Annoying by Hiding Group Elements". In: *Computer Security - ESORICS 2023 - 28th European Symposium on Research in Computer Security, The Hague, The Netherlands, September 25-29, 2023, Proceedings, Part I*. Ed. by Gene Tsudik, Mauro Conti, Kaitai Liang, and Georgios Smaragdakis. Vol. 14344. Lecture Notes in Computer Science. Springer, 2023, pp. 168–188. DOI: 10.1007/978-3-031-50594-2_9. URL: https://doi.org/10.1007/978-3-031-50594-2_9 (cit. on pp. 15, 17, 141, 143, 175).

[TMM24a] Marcel Tiepelt, Christian Martin, and Nils Maeurer. *Post-Quantum Ready Key Agreement for Aviation*. Cryptology ePrint Archive, Paper 2024/1096. https://eprint.iacr.org/2024/1096. 2024. DOI: 10.62056/aebn2isfg. URL: https://eprint.iacr.org/2024/1096 (cit. on pp. 14, 17, 143, 155, 166, 167).

[TMM24b] Marcel Tiepelt, Christian Martin, and Nils Mäurer. "Post-Quantum Ready Key Agreement for Aviation". In: *IACR Communications in Cryptology* 1.1 (Apr. 9, 2024). ISSN: 3006-5496. DOI: 10.62056/aebn2isfg. URL: https://doi.org/10.62056/aebn2isfg (cit. on pp. 14, 17, 140, 143, 155).

[TS19] Marcel Tiepelt and Alan Szepieniec. "Quantum LLL with an Application to Mersenne Number Cryptosystems". In: *Progress in Cryptology – LATINCRYPT 2019*. Ed. by Peter Schwabe and Nicolas Thériault. Cham: Springer International Publishing, 2019, pp. 3–23. ISBN: 978-3-030-30530-7. DOI: 10.1007/978-3-030-30530-7_1. URL: https://doi.org/10.1007/978-3-030-30530-7_1 (cit. on pp. 16, 51, 61, 78).

[WVW92] Diffie Whitfield, Paul C. Van Oorshot, and Michael J. Wiener. “Authentication and authenticated key exchanges”. In: *Designs, Codes and Cryptography* (1992). DOI: 10.1007/BF00124891 (cit. on p. 156).

[Zal99] Christof Zalka. “Grover’s quantum searching algorithm is optimal”. In: *Physical Review A* 60.4 (1999), p. 2746 (cit. on p. 44).

Acronyms

AES	Advanced Encryption Standard. 28, 44, 53
aPAKE	asymmetric Password Authenticated Key Exchange. 176, 178, 179, 187
BDD	Bounded Distance Decoding. 60, 61
BQP	Bounded Error Quantum Polynomial Time. 7
BSI	German Federal Office for Information Security. 7
CDH	Computational Diffie–Hellman. 21
DDH	Decisional Diffie–Hellman. 21, 140, 141, 165
DFS	depth-first search. 34, 37
DH	Diffie–Hellman. 139, 155–157, 159, 163
DLOG	Discrete Logarithm. 139, 140
ECC	Error Correcting Code. 48
EUF-CMA	Existential Unforgeability under Chosen Message Attack. 21, 22, 157, 158, 162, 164, 191
FORS	Forest of Random Subsets. 11, 29, 52, 54, 56, 58, 89, 91
HW	Hamming weight. 47
IC	Ideal Cipher. 22
IKEv2	Internet Key Exchange Version 2. 156, 157, 163
IND-CCA	Indistinguishability under Chosen Ciphertext Attack. 10, 15, 20, 21, 28, 48, 65, 135
IND-CPA	Indistinguishability under Chosen Plaintext Attack. 20, 21, 157, 158, 162, 165, 191
KEM	Key Encapsulation Mechanism. 9, 20, 21, 27, 28, 30, 44, 47, 48, 67, 68, 71, 106, 124, 139, 140, 155, 157–160, 162, 165, 191
KH-AKE	Key-Hiding Authenticated Key Exchange. 176, 177
LDACS	L-band Digital Aeronautic Communication System. 139, 140, 191
LHC	Mersenne Low Hamming Combination. 47–49
LLL	Lenstra-Lenstra-Lovász. 16, 49, 61
LWE	Learning with Errors. 60, 105
MAC	Message Authentication Code. 156, 158, 162, 163
NIST	National Institute for Standards and Technology. 7, 9, 15, 27–30, 37, 40, 44, 47, 51, 58, 65, 83, 106, 124, 135, 139, 156, 157

OTS One-Time-Signature. 51
OWF One-Way Function. 23

PAKE Password Authenticated Key Exchange. 14, 140, 141, 149, 175–179
PPT Probabilistic Polynomial Time. 20, 21
PRF Pseudo Random Function. 22, 163

QAA Quantum Amplitude Amplification. 29, 33, 83, 90, 92, 94, 96, 97
QPE Quantum Phase Estimation. 34–36, 108–111
QRACM Quantum Accessible Random Access Memory. 30, 33, 106, 115

SHA Secure Hash Algorithm. 28, 44, 53
SIGMA SIGn-and-MAc. 156, 157, 159, 162
SVP Shortest Vector Problem. 59–62, 105

UUF Universal Unforgability. 22

WOTS Winternitz One Time Signature. 11, 29, 52, 53, 56, 89, 93, 96

XMSS eXtended Merle Signature Scheme. 11, 29, 51, 52, 55, 56, 89, 93, 95

List of Figures

List of Tables

Part IV

APPENDIX

A

Complete Results of Decryption Failure Attack

Here, we report the precise results from the decryption failure attack in Chapter 5. Table A.1 shows the complete list of empirical results of our attack for multiple secret-keys. For a growing number of decryption failures the number of *correct* intervals decreases. However, the number of position of ones in these intervals increases. Moreover the width of the *empty* spaces increases, which improves the success probability of the attack. The size of the set $\mathtt{I}_{sample}$ is significantly lower than the number of enclosed positions of ones, suggesting that each range encloses multiple secret bit positions.

TABLE A.1: Experimental results of our implementation where the column *correct* denotes the number of correct intervals detected, and *sample* the number of intervals that require sampling.

seed	decryption failures	*#correct*	*#sample*	#secret ones in *correct*	$\mathbb{E}[\lvert\beta_{\cdot,\cdot}\rvert]$	$\mathbb{E}[l'']$	$\mathbb{E}[l]$	$\mathbb{E}[\mathbb{P}[success]]$ per part	#Grover
$c14532$	2^9	144	54	125	5036	2240	959	0.458	2^{74}
$c14532$	2^{10}	112	35	164	5019	2315	1167	0.4654	2^{51}
$c14532$	2^{11}	92	28	174	5467	2743	1249	0.4655	2^{46}
$c14532$	2^{12}	89	29	176	5098	2542	1206	0.4622	2^{45}
$195bc2$	2^9	131	47	142	5140	2408	1325	0.4689	2^{62}
$195bc2$	2^{10}	111	33	172	5078	2405	1238	0.4679	2^{46}
$195bc2$	2^{11}	98	33	166	5261	2487	1386	0.464	2^{50}
$195bc2$	2^{12}	91	33	168	5182	2482	1422	0.4695	2^{48}
$0d86a4$	2^9	144	53	136	4506	2256	1113	0.4729	2^{65}
$0d86a4$	2^{10}	111	43	155	4620	2580	1137	0.4869	2^{53}
$0d86a4$	2^{11}	90	42	154	4760	2924	1261	0.4942	2^{52}
$0d86a4$	2^{12}	89	42	154	4650	2971	1118	0.4924	2^{52}
170784	2^9	134	53	120	5140	2726	1268	0.4905	2^{70}
170784	2^{10}	117	41	151	4884	2502	1346	0.4878	2^{55}
170784	2^{11}	101	32	171	4962	2837	1584	0.5049	2^{42}
170784	2^{12}	102	30	177	4835	2822	1676	0.5042	2^{39}

B

Further Results for Quantum Enumeration

B.1 Results from Lower Bounds

B.1.1 *Tables for the Quasi-Sqrt and the Canonical Bit Security*

As an alternative to the comparison with the cost of Grover on AES one can check whether the attack cost under depth constraints ever achieves a "quasi-quadratic" speedup over classical enumeration (meaning going from $\#\mathcal{T}$ gates to $\sqrt{\#\mathcal{T} \cdot h}$, where h is the height of $\mathcal{T}$), as it could be expected from Theorem 2. Alternatively, for a scheme like Kyber-512 (with analogous notions for -768 and -1024), one could consider an attack successful if its gate cost is lower than $2^{\text{canonical bit security}} = 2^{128}$, albeit this is explicitly not the cost metric chosen by NIST and plausibly targeted by the Kyber team. In Tables B.1 and B.2 we present the results for these two metrics which are computed as specified in Section 7.3.

B.1.2 *Figures for the Query-based Result*

In this section we present the figures from our estimations of the cost of quantum enumeration that we have not reported in Section 7.3. For all cost estimations the quantum operator GCost$(\mathcal{W})$ and T-Depth$(\mathcal{W})$ are estimated as in Section 7.2.1 and with $DF(\mathcal{W})$, $QD(\mathcal{W})$, $WQ(\mathcal{T}, W)$ as in Section 7.1,

- Figures B.1 and B.2 show the cost estimation for Kyber-512 and Kyber-768 with T-Depth(QPE$(\mathcal{W})) \leq \{2^{40},\ 2^{64},\ 2^{96}\}$.

B.1.3 *Figures for the Circuit-based Results*

Next we present additional figures of quantum enumeration cost estimation to the ones in Section 7.3. As before, GCost$(\mathcal{W})$ and TDepth$(\mathcal{W})$ are estimated as in Section 7.2.2 and $DF(\mathcal{W})$, $QD(\mathcal{W})$, $WQ(\mathcal{T}, W)$ as in Section 7.1,

- Figure B.3 shows the cost estimation without any MaxDepth constraint on T-Depth(QPE$(\mathcal{W})$).
- Figures B.4 to B.6 show the cost estimation for Kyber-512, -768 and Kyber-1024 assuming T-Depth(QPE$(\mathcal{W})) \leq \{2^{40},\ 2^{64},\ 2^{96}\}$.

TABLE B.1: Summary of the values for the Jensen's gap 2^z at crossover points of our combined classical-quantum enumeration attacks against Kyber and the quasi-square root speed up $\sqrt{\#\mathcal{T}\cdot n}$. We remark that exact crossovers happen at fractional values of z. In this table we round down threshold values of z. MAXDEPTH is abbreviated to MD. Cost is as in Table 7.6.

less likely to be feasible — more likely to be feasible

Crossover points when comparing quasi-square-root against $\log \mathbb{E}[\text{Quantum GCOST}]$ (cf. Equation (7.7)) with …

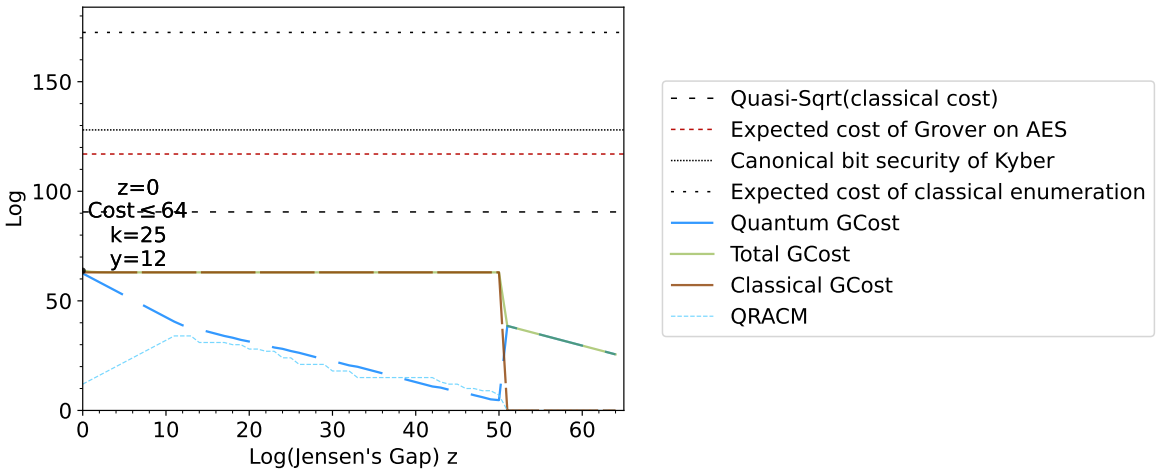
		…$\mathcal{W}$ as in Section 7.2.1			…$\mathcal{W}$ as in Section 7.2.2		
MD	Kyber	LB/UB	UB/UB	LB/LB	LB/UB	UB/UB	LB/LB
2^{40}	-512	$z \geq 0, k \leq 25$ Cost $\geq 2^{63}$	$z \geq 29, k \leq 11$ Cost $\geq 2^{98}$	$z \geq 25, k \leq 59$ Cost $\geq 2^{89}$	$z \geq 2, k \leq 24$ Cost $\geq 2^{90}$	$z \geq 45, k \leq 12$ Cost $\geq 2^{98}$	$z \geq 41, k \leq 63$ Cost $\geq 2^{89}$
	-768	$z \geq 8, k \leq 75$ Cost $\geq 2^{166}$	$z > 64$	$z \geq 63, k \leq 77$ Cost $\geq 2^{165}$	$z \geq 25, k \leq 67$ Cost $\geq 2^{164}$	$z > 64$	$z > 64$
	-1024	$z \geq 41, k \leq 115$ Cost $\geq 2^{261}$	$z > 64$	$z > 64$	$z \geq 57, k \leq 105$ Cost $\geq 2^{262}$	$z > 64$	$z > 64$
2^{64}	-512	$z \geq 0, k \leq 26$ Cost $\geq 2^{63}$	$z \geq 17, k \leq 11$ Cost $\geq 2^{98}$	$z \geq 13, k \leq 59$ Cost $\geq 2^{89}$	$z \geq 0, k \leq 26$ Cost $\geq 2^{75}$	$z \geq 33, k \leq 5$ Cost $\geq 2^{98}$	$z \geq 29, k \leq 54$ Cost $\geq 2^{89}$
	-768	$z \geq 0, k \leq 64$ Cost $\geq 2^{157}$	$z \geq 54, k \leq 33$ Cost $\geq 2^{170}$	$z \geq 51, k \leq 77$ Cost $\geq 2^{165}$	$z \geq 13, k \leq 67$ Cost $\geq 2^{164}$	$z > 64$	$z > 64$
	-1024	$z \geq 28, k \leq 105$ Cost $\geq 2^{262}$	$z > 64$	$z > 64$	$z \geq 45, k \leq 100$ Cost $\geq 2^{262}$	$z > 64$	$z > 64$
2^{96}	-512	$z \geq 0, k \leq 26$ Cost $\geq 2^{63}$	$z \geq 1, k \leq 2$ Cost $\geq 2^{98}$	$z \geq 0, k \leq 40$ Cost $\geq 2^{89}$	$z \geq 0, k \leq 26$ Cost $\geq 2^{75}$	$z \geq 24, k \leq 1$ Cost $\geq 2^{98}$	$z \geq 24, k \leq 40$ Cost $\geq 2^{89}$
	-768	$z \geq 0, k \leq 53$ Cost $\geq 2^{126}$	$z \geq 38, k \leq 12$ Cost $\geq 2^{170}$	$z \geq 35, k \leq 77$ Cost $\geq 2^{165}$	$z \geq 0, k \leq 64$ Cost $\geq 2^{158}$	$z \geq 54, k \leq 12$ Cost $\geq 2^{171}$	$z \geq 52, k \leq 62$ Cost $\geq 2^{164}$
	-1024	$z \geq 12, k \leq 100$ Cost $\geq 2^{262}$	$z > 64$	$z > 64$	$z \geq 29, k \leq 100$ Cost $\geq 2^{262}$	$z > 64$	$z > 64$
∞	-512	$z \geq 0, k \leq 26$ Cost $\geq 2^{63}$	$z \geq 0, k \leq 1$ Cost $\geq 2^{97}$	$z \geq 0, k \leq 40$ Cost $\geq 2^{89}$	$z \geq 0, k \leq 26$ Cost $\geq 2^{75}$	$z \geq 24, k \leq 1$ Cost $\geq 2^{98}$	$z \geq 24, k \leq 40$ Cost $\geq 2^{89}$
	-768	$z \geq 0, k \leq 37$ Cost $\geq 2^{113}$	$z \geq 2, k \leq 3$ Cost $\geq 2^{171}$	$z \geq 0, k \leq 31$ Cost $\geq 2^{165}$	$z \geq 0, k \leq 37$ Cost $\geq 2^{138}$	$z \geq 28, k \leq 3$ Cost $\geq 2^{170}$	$z \geq 25, k \leq 31$ Cost $\geq 2^{165}$
	-1024	$z \geq 0, k \leq 33$ Cost $\geq 2^{206}$	$z \geq 2, k \leq 3$ Cost $\geq 2^{269}$	$z \geq 0, k \leq 1$ Cost $\geq 2^{262}$	$z \geq 0, k \leq 33$ Cost $\geq 2^{232}$	$z \geq 29, k \leq 3$ Cost $\geq 2^{268}$	$z \geq 26, k \leq 11$ Cost $\geq 2^{263}$
$\infty_{k=0}$	-512	$z \geq 0, k = 0$ Cost $\geq 2^{89}$	$z \geq 0, k = 0$ Cost $\geq 2^{97}$	$z \geq 0, k = 0$ Cost $\geq 2^{89}$	$z \geq 24, k = 0$ Cost $\geq 2^{90}$	$z \geq 24, k = 0$ Cost $\geq 2^{98}$	$z \geq 24, k = 0$ Cost $\geq 2^{90}$
	-768	$z \geq 0, k = 0$ Cost $\geq 2^{165}$	$z \geq 0, k = 0$ Cost $\geq 2^{170}$	$z \geq 0, k = 0$ Cost $\geq 2^{165}$	$z \geq 25, k = 0$ Cost $\geq 2^{165}$	$z \geq 25, k = 0$ Cost $\geq 2^{171}$	$z \geq 25, k = 0$ Cost $\geq 2^{165}$
	-1024	$z \geq 0, k = 0$ Cost $\geq 2^{262}$	$z \geq 0, k = 0$ Cost $\geq 2^{268}$	$z \geq 0, k = 0$ Cost $\geq 2^{262}$	$z \geq 26, k = 0$ Cost $\geq 2^{263}$	$z \geq 26, k = 0$ Cost $\geq 2^{269}$	$z \geq 26, k = 0$ Cost $\geq 2^{263}$

TABLE B.2: Summary of the values for the Jensen's gap 2^z at crossover points of our combined classical-quantum enumeration attacks against Kyber and the canonical 128, 192, 256 bit security respectively. We remark that exact crossovers happen at fractional values of z. In this table we round down threshold values of z. MAXDEPTH is abbreviated to MD. Cost is as in Table 7.6

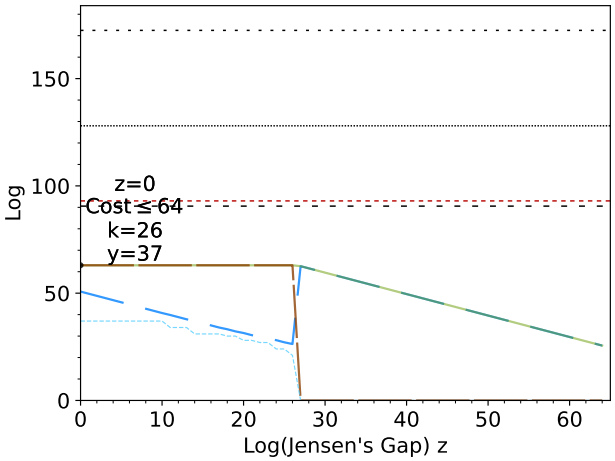
less likely to be feasible — more likely to be feasible

Crossover points when comparing quasi-square-root against $\log \mathbb{E}[\text{Quantum GCOST}]$ (cf. Equation (7.7)) with …

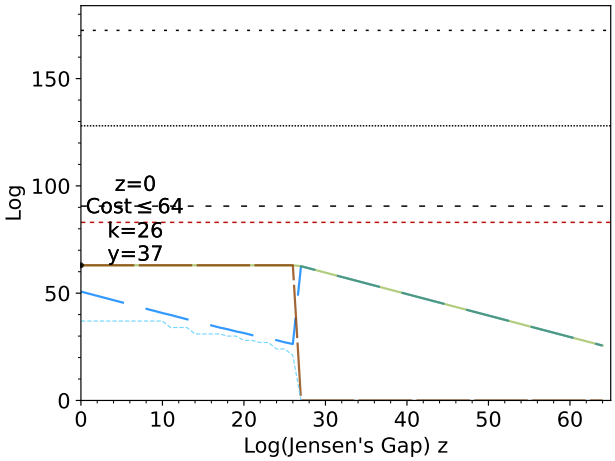
		…$\mathcal{W}$ as in Section 7.2.1			…$\mathcal{W}$ as in Section 7.2.2		
MD	Kyber	LB/UB	UB/UB	LB/LB	LB/UB	UB/UB	LB/LB
2^{40}	-512	$z \geq 0, k \leq 25$ Cost $\geq 2^{63}$	$z \geq 15, k \leq 28$ Cost $\geq 2^{126}$	$z \geq 6, k \leq 92$ Cost $\geq 2^{127}$	$z \geq 0, k \leq 27$ Cost $\geq 2^{94}$	$z \geq 30, k \leq 24$ Cost $\geq 2^{127}$	$z \geq 22, k \leq 96$ Cost $\geq 2^{126}$
	-768	$z \geq 0, k \leq 87$ Cost $\geq 2^{184}$	$z \geq 56, k \leq 33$ Cost $\geq 2^{190}$	$z \geq 50, k \leq 114$ Cost $\geq 2^{191}$	$z \geq 12, k \leq 80$ Cost $\geq 2^{191}$	$z > 64$	$z > 64$
	-1024	$z \geq 44, k \leq 112$ Cost $\geq 2^{255}$	$z > 64$	$z > 64$	$z \geq 61, k \leq 105$ Cost $\geq 2^{254}$	$z > 64$	$z > 64$
2^{64}	-512	$z \geq 0, k \leq 26$ Cost $\geq 2^{63}$	$z \geq 3, k \leq 28$ Cost $\geq 2^{126}$	$z \geq 0, k \leq 83$ Cost $\geq 2^{115}$	$z \geq 0, k \leq 26$ Cost $\geq 2^{75}$	$z \geq 18, k \leq 24$ Cost $\geq 2^{127}$	$z \geq 10, k \leq 79$ Cost $\geq 2^{127}$
	-768	$z \geq 0, k \leq 64$ Cost $\geq 2^{157}$	$z \geq 44, k \leq 33$ Cost $\geq 2^{190}$	$z \geq 38, k \leq 114$ Cost $\geq 2^{191}$	$z \geq 0, k \leq 67$ Cost $\geq 2^{190}$	$z \geq 60, k \leq 37$ Cost $\geq 2^{191}$	$z \geq 54, k \leq 106$ Cost $\geq 2^{191}$
	-1024	$z \geq 32, k \leq 100$ Cost $\geq 2^{254}$	$z > 64$	$z > 64$	$z \geq 49, k \leq 100$ Cost $\geq 2^{254}$	$z > 64$	$z > 64$
2^{96}	-512	$z \geq 0, k \leq 26$ Cost $\geq 2^{63}$	$z \geq 0, k \leq 2$ Cost $\geq 2^{100}$	$z \geq 0, k \leq 40$ Cost $\geq 2^{89}$	$z \geq 0, k \leq 26$ Cost $\geq 2^{75}$	$z \geq 3, k \leq 5$ Cost $\geq 2^{126}$	$z \geq 0, k \leq 46$ Cost $\geq 2^{115}$
	-768	$z \geq 0, k \leq 53$ Cost $\geq 2^{126}$	$z \geq 28, k \leq 33$ Cost $\geq 2^{190}$	$z \geq 22, k \leq 77$ Cost $\geq 2^{191}$	$z \geq 0, k \leq 64$ Cost $\geq 2^{158}$	$z \geq 44, k \leq 26$ Cost $\geq 2^{191}$	$z \geq 39, k \leq 77$ Cost $\geq 2^{190}$
	-1024	$z \geq 16, k \leq 100$ Cost $\geq 2^{254}$	$z > 64$	$z > 64$	$z \geq 33, k \leq 100$ Cost $\geq 2^{254}$	$z > 64$	$z > 64$
∞	-512	$z \geq 0, k \leq 26$ Cost $\geq 2^{63}$	$z \geq 0, k \leq 1$ Cost $\geq 2^{97}$	$z \geq 0, k \leq 40$ Cost $\geq 2^{89}$	$z \geq 0, k \leq 26$ Cost $\geq 2^{75}$	$z \geq 0, k \leq 1$ Cost $\geq 2^{122}$	$z \geq 0, k \leq 40$ Cost $\geq 2^{113}$
	-768	$z \geq 0, k \leq 37$ Cost $\geq 2^{113}$	$z \geq 0, k \leq 3$ Cost $\geq 2^{173}$	$z \geq 0, k \leq 31$ Cost $\geq 2^{165}$	$z \geq 0, k \leq 37$ Cost $\geq 2^{138}$	$z \geq 7, k \leq 3$ Cost $\geq 2^{191}$	$z \geq 0, k \leq 31$ Cost $\geq 2^{190}$
	-1024	$z \geq 0, k \leq 33$ Cost $\geq 2^{206}$	$z \geq 16, k \leq 3$ Cost $\geq 2^{255}$	$z \geq 7, k \leq 1$ Cost $\geq 2^{255}$	$z \geq 0, k \leq 33$ Cost $\geq 2^{232}$	$z \geq 42, k \leq 3$ Cost $\geq 2^{255}$	$z \geq 34, k \leq 11$ Cost $\geq 2^{255}$
$\infty_{k=0}$	-512	$z \geq 0, k = 0$ Cost $\geq 2^{89}$	$z \geq 0, k = 0$ Cost $\geq 2^{97}$	$z \geq 0, k = 0$ Cost $\geq 2^{89}$	$z \geq 0, k = 0$ Cost $\geq 2^{114}$	$z \geq 0, k = 0$ Cost $\geq 2^{122}$	$z \geq 0, k = 0$ Cost $\geq 2^{114}$
	-768	$z \geq 0, k = 0$ Cost $\geq 2^{165}$	$z \geq 0, k = 0$ Cost $\geq 2^{170}$	$z \geq 0, k = 0$ Cost $\geq 2^{165}$	$z \geq 0, k = 0$ Cost $\geq 2^{190}$	$z \geq 5, k = 0$ Cost $\geq 2^{191}$	$z \geq 0, k = 0$ Cost $\geq 2^{190}$
	-1024	$z \geq 7, k = 0$ Cost $\geq 2^{255}$	$z \geq 13, k = 0$ Cost $\geq 2^{255}$	$z \geq 7, k = 0$ Cost $\geq 2^{255}$	$z \geq 34, k = 0$ Cost $\geq 2^{255}$	$z \geq 40, k = 0$ Cost $\geq 2^{255}$	$z \geq 34, k = 0$ Cost $\geq 2^{255}$



(a) Kyber-512, MaxDepth $= 2^{40}$

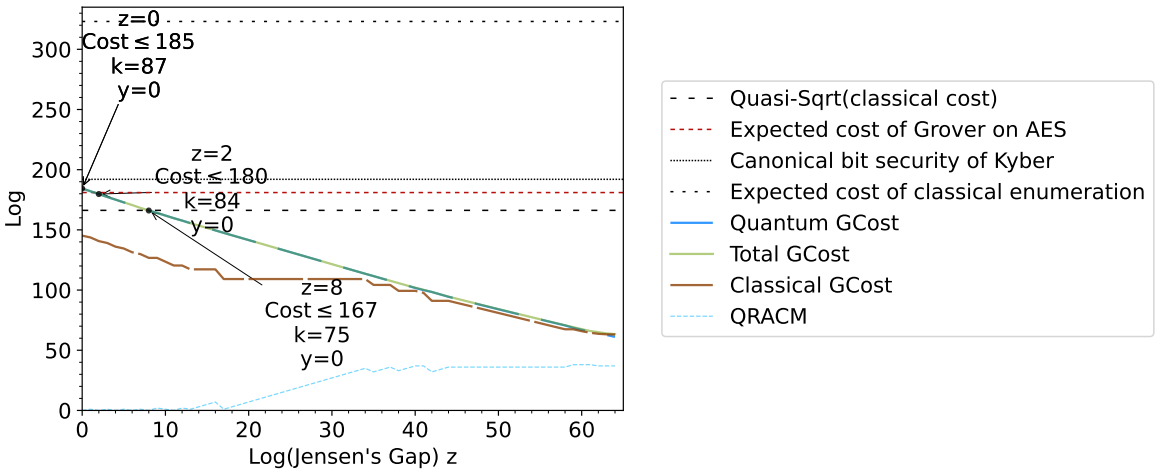


(b) Kyber-512, MaxDepth $= 2^{64}$

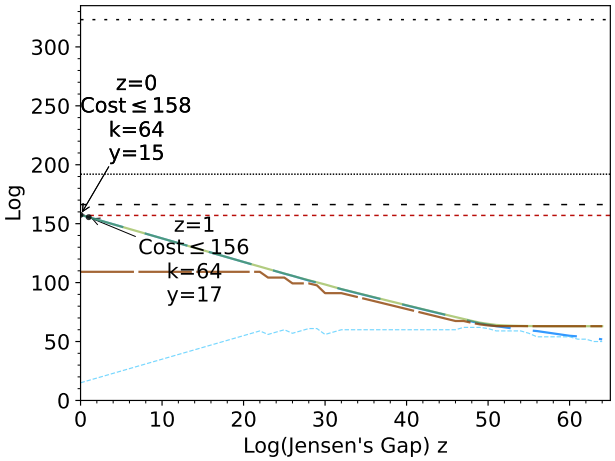


(c) Kyber-512, MaxDepth $= 2^{96}$

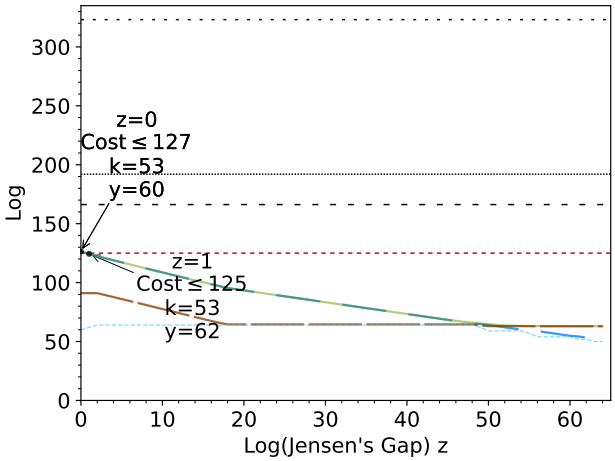
FIGURE B.1: Cost estimation for Kyber-512 under different MaxDepthrestrictions with the instantiation for operator $\mathcal{W}$ as in Section 7.2.1 and with DF $= 1$, QD $= 1$, $b = 1$ (see Section 7.1), and corresponding to the lower bound (LB/UB) for Conjecture 3.



(a) Kyber-768, MaxDepth $= 2^{40}$.

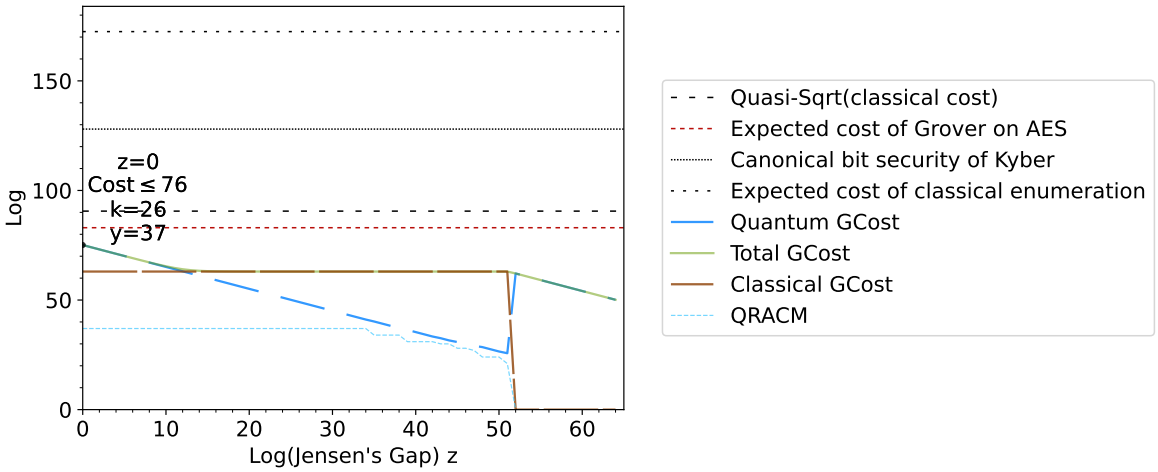


(b) Kyber-768, MaxDepth $= 2^{64}$.

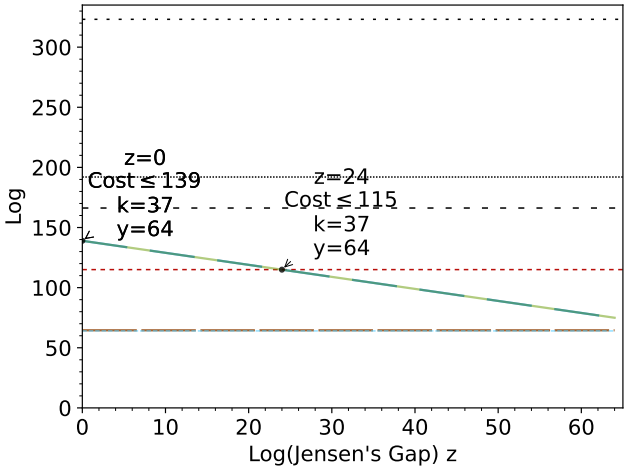


(c) Kyber-768, MaxDepth $= 2^{96}$.

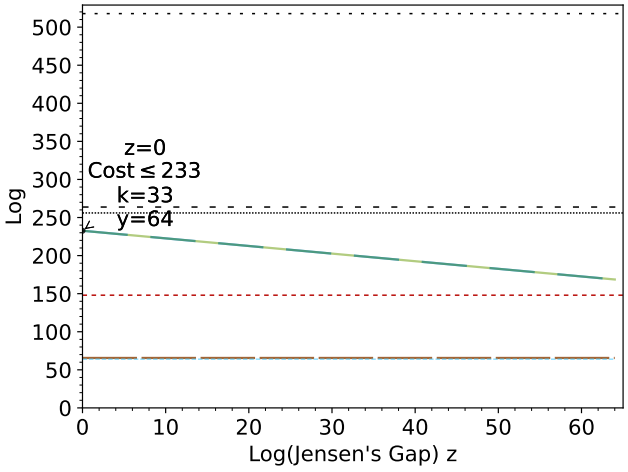
Figure B.2: Cost estimation for Kyber-768 under different MaxDepthrestrictions with the instantiation for operator $\mathcal{W}$ as in Section 7.2.1 and with DF $= 1$, QD $= 1$, $b = 1$ (see Section 7.1), and corresponding to the lower bound (LB/UB) for Conjecture 3.



(a) Cost estimation for Kyber-512 w/ MaxDepth $= \infty$.

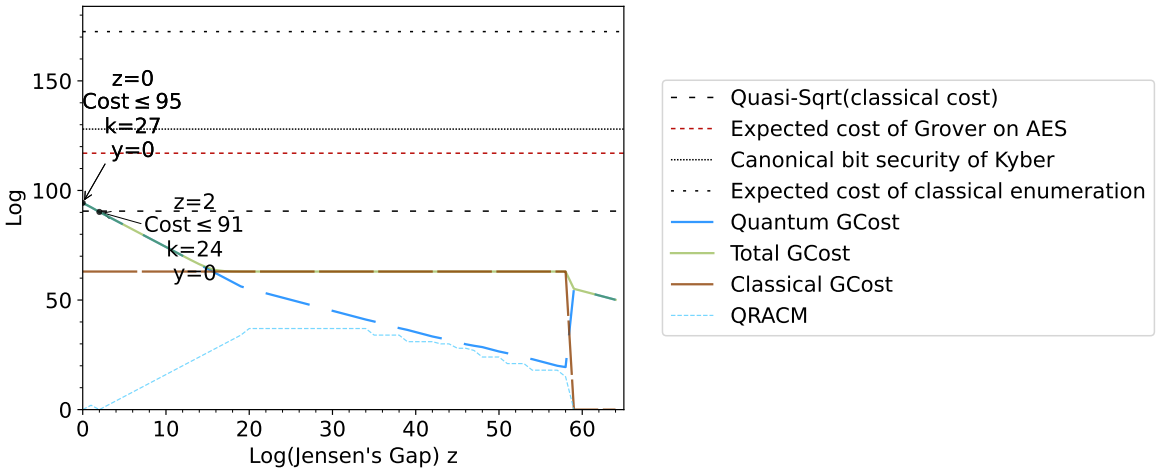


(b) Cost estimation for Kyber-768 w/ MaxDepth $= \infty$.

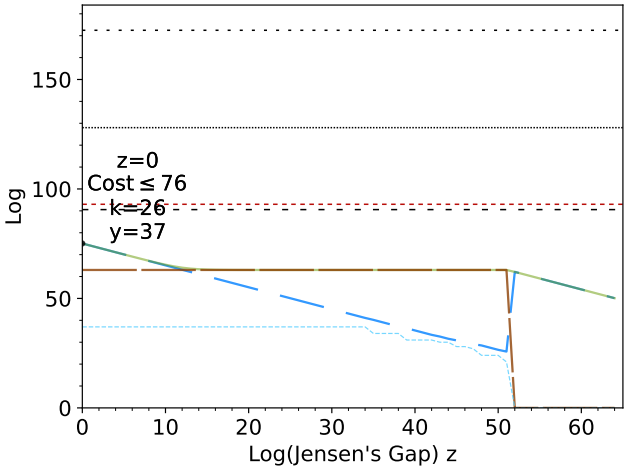


(c) Cost estimation for Kyber-1024 w/ MaxDepth $= \infty$.

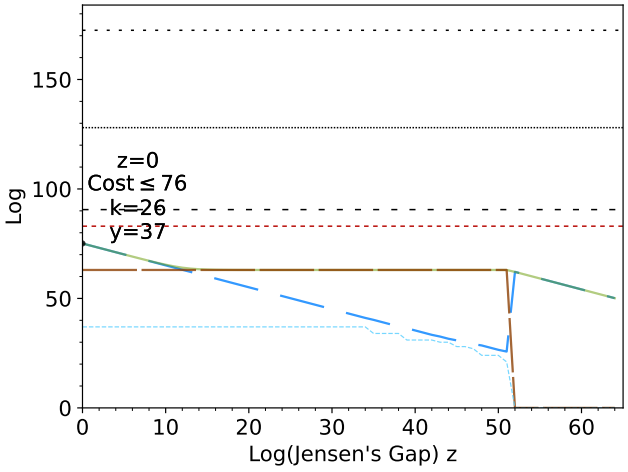
Figure B.3: Cost estimation for Kyber without MaxDepth restriction and for operator $\mathcal{W}$ as in Section 7.2.2 and with DF $= 1$, QD $= 1$, $b = 1$, and corresponding to the lower bound (LB/UB) for Conjecture 3.



(a) Kyber-512, MaxDepth $= 2^{40}$

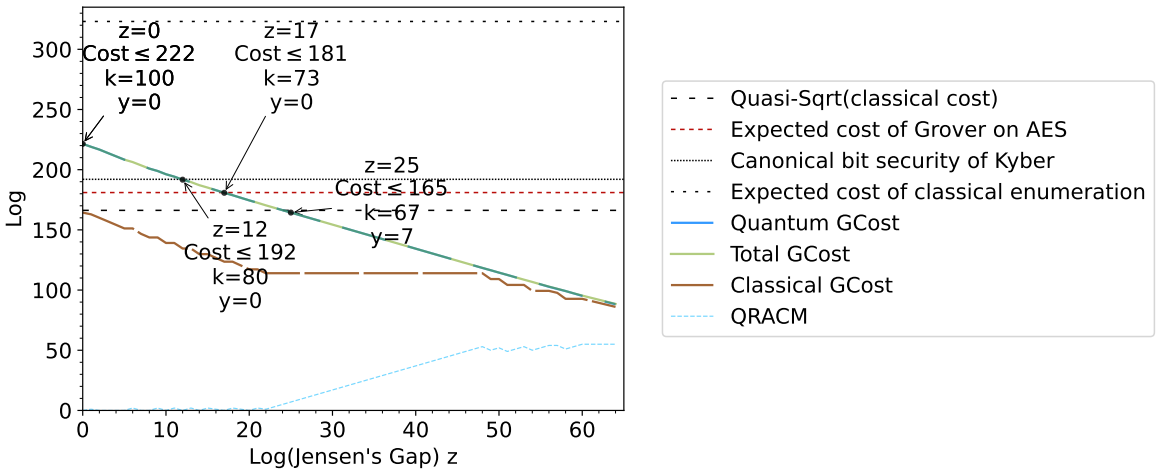


(b) Kyber-512, MaxDepth $= 2^{64}$

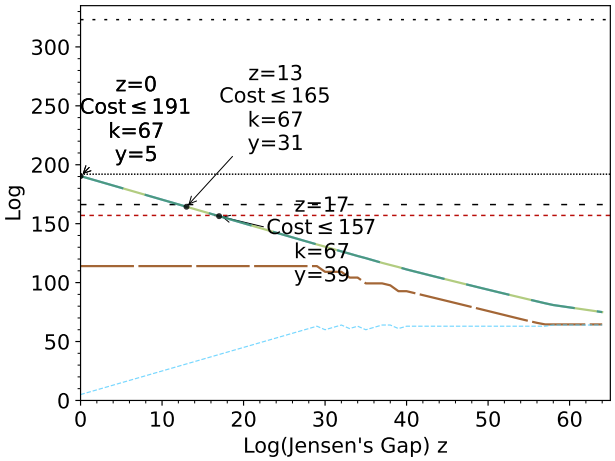


(c) Kyber-512, MaxDepth $= 2^{96}$

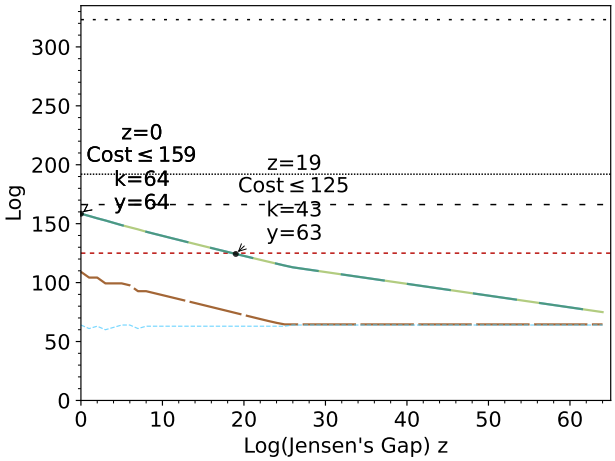
Figure B.4: Cost estimation for Kyber-512 under different MaxDepthrestrictions with operator $\mathcal{W}$ as in Section 7.2.2 and with DF $= 1$, QD $= 1$, $b = 1$, and corresponding to the lower bound (LB/UB) for Conjecture 3.



(a) Kyber-768, MaxDepth $= 2^{40}$

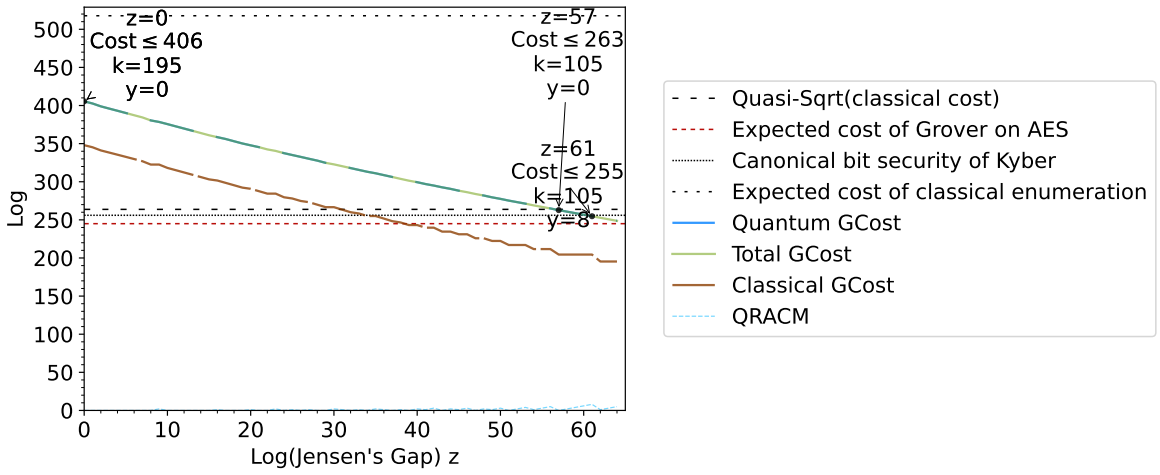


(b) Kyber-768, MaxDepth $= 2^{64}$

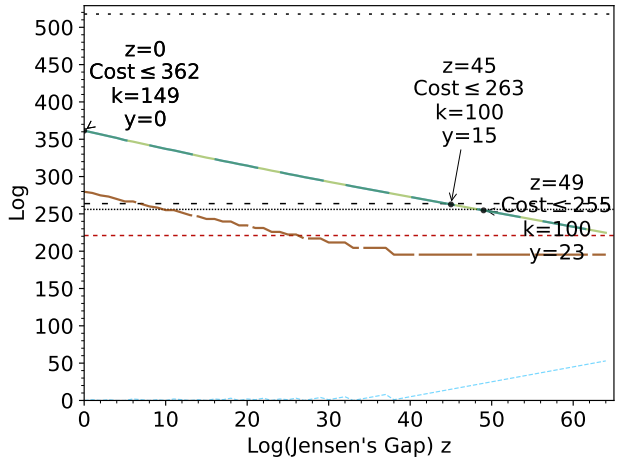


(c) Kyber-768, MaxDepth $= 2^{96}$

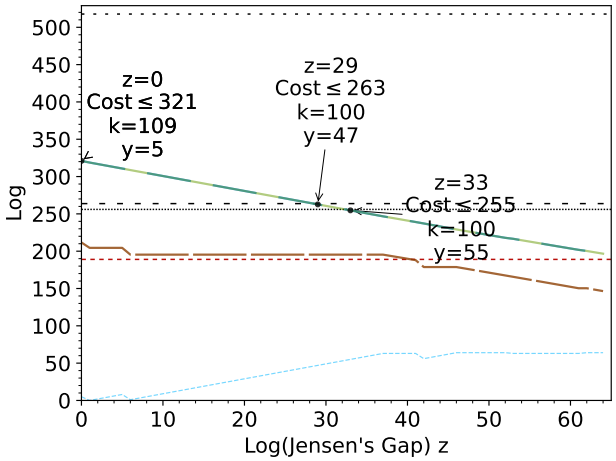
Figure B.5: Cost estimation for Kyber-768 under different MaxDepthrestrictions with the instantiation for operator $\mathcal{W}$ as in Section 7.2.2 and with DF $= 1$, QD $= 1$, $b = 1$ (see Section 7.1), and corresponding to the lower bound (LB/UB) for Conjecture 3.



(a) Kyber-1024, MaxDepth $= 2^{40}$.



(b) Kyber-1024, MaxDepth $= 2^{64}$.



(c) Kyber-1024, MaxDepth $= 2^{96}$.

Figure B.6: Cost estimation for Kyber-1024 under different MaxDepthrestrictions with the instantiation for operator $\mathcal{W}$ as in Section 7.2.2 and with DF $= 1$, QD $= 1$, $b = 1$ (see Section 7.1), and corresponding to the lower bound (LB/UB) for Conjecture 3.

B.2 Results beyond Lower Bounds

We replicate the analysis performed in Section 7.3.2, estimating the cost of a combined classical-quantum attack as described in Sec. 7.1.3 and 7.1.4. In Tables B.3 and B.4 we summarize the values for the Jensen's gap 2^z at crossover points of our combined classical-quantum enumeration attacks against the quasi-square-root and the canonical $128, 192, 256$ bit security of Kyber.

The tables corresponds to attacks in the settings for $\mathcal{W}$ as in Section 7.2.1 and Section 7.2.2 with $\mathcal{C} = 2$, $\varepsilon = 20$, $b = 1/64$, resulting in $\mathrm{DF}(\mathcal{W}) = n \log \mathcal{C} = n$ and $\mathrm{QD}(\mathcal{W}) = \lceil 20 \cdot \log(1/\delta_{DMV}) \rceil = \lceil 20 \cdot \log(n) \rceil$, and $\mathrm{WQ}(\mathcal{T}, \mathcal{W}) = 64\sqrt{\#\mathcal{T}h}$. We omit the plots for the alternative bounds.

TABLE B.3: Summary of the values for the Jensen's gap 2^z at crossover points of our combined classical-quantum enumeration attacks against Kyber and the quasi-square root speed up $\sqrt{n\#\mathcal{T}}$. We remark that exact crossovers happen at fractional values of z. In this table we round down threshold values of z. MAXDEPTH is abbreviated to MD. The results are estimated from the bounds $\mathcal{C} = 2$, $\varepsilon = 20$, $b = 1/64$.

less likely to be feasible — more likely to be feasible

Crossover points when comparing quasi-square-root against $\log \mathbb{E}[\text{Quantum GCOST}]$ (cf. Equation (7.7)) with …

		…$\mathcal{W}$ as in Section 7.2.1			…$\mathcal{W}$ as in Section 7.2.2		
MD	Kyber	LB/UB	UB/UB	LB/LB	LB/UB	UB/UB	LB/LB
2^{40}	-512	$z \geq 0, k \leq 25$ Cost $\geq 2^{90}$	$z \geq 43, k \leq 11$ Cost $\geq 2^{98}$	$z \geq 39, k \leq 59$ Cost $\geq 2^{89}$	$z \geq 16, k \leq 24$ Cost $\geq 2^{90}$	$z \geq 59, k \leq 8$ Cost $\geq 2^{98}$	$z \geq 55, k \leq 58$ Cost $\geq 2^{89}$
	-768	$z \geq 22, k \leq 69$ Cost $\geq 2^{166}$	$z > 64$	$z > 64$	$z \geq 39, k \leq 67$ Cost $\geq 2^{165}$	$z > 64$	$z > 64$
	-1024	$z \geq 55, k \leq 109$ Cost $\geq 2^{262}$	$z > 64$	$z > 64$	$z > 64$	$z > 64$	$z > 64$
2^{64}	-512	$z \geq 0, k \leq 26$ Cost $\geq 2^{72}$	$z \geq 31, k \leq 4$ Cost $\geq 2^{98}$	$z \geq 27, k \leq 55$ Cost $\geq 2^{89}$	$z \geq 7, k \leq 26$ Cost $\geq 2^{90}$	$z \geq 47, k \leq 1$ Cost $\geq 2^{98}$	$z \geq 46, k \leq 40$ Cost $\geq 2^{89}$
	-768	$z \geq 10, k \leq 64$ Cost $\geq 2^{166}$	$z > 64$	$z > 64$	$z \geq 27, k \leq 67$ Cost $\geq 2^{165}$	$z > 64$	$z > 64$
	-1024	$z \geq 43, k \leq 100$ Cost $\geq 2^{262}$	$z > 64$	$z > 64$	$z \geq 60, k \leq 100$ Cost $\geq 2^{261}$	$z > 64$	$z > 64$
2^{96}	-512	$z \geq 0, k \leq 26$ Cost $\geq 2^{72}$	$z \geq 22, k \leq 1$ Cost $\geq 2^{97}$	$z \geq 21, k \leq 40$ Cost $\geq 2^{90}$	$z \geq 7, k \leq 26$ Cost $\geq 2^{90}$	$z \geq 46, k \leq 1$ Cost $\geq 2^{98}$	$z \geq 46, k \leq 40$ Cost $\geq 2^{89}$
	-768	$z \geq 0, k \leq 61$ Cost $\geq 2^{154}$	$z \geq 52, k \leq 12$ Cost $\geq 2^{171}$	$z \geq 49, k \leq 62$ Cost $\geq 2^{165}$	$z \geq 11, k \leq 58$ Cost $\geq 2^{165}$	$z > 64$	$z > 64$
	-1024	$z \geq 27, k \leq 100$ Cost $\geq 2^{262}$	$z > 64$	$z > 64$	$z \geq 44, k \leq 99$ Cost $\geq 2^{261}$	$z > 64$	$z > 64$
∞	-512	$z \geq 0, k \leq 26$ Cost $\geq 2^{72}$	$z \geq 22, k \leq 1$ Cost $\geq 2^{97}$	$z \geq 21, k \leq 40$ Cost $\geq 2^{90}$	$z \geq 7, k \leq 26$ Cost $\geq 2^{90}$	$z \geq 46, k \leq 1$ Cost $\geq 2^{98}$	$z \geq 46, k \leq 40$ Cost $\geq 2^{89}$
	-768	$z \geq 0, k \leq 37$ Cost $\geq 2^{136}$	$z \geq 25, k \leq 3$ Cost $\geq 2^{170}$	$z \geq 22, k \leq 31$ Cost $\geq 2^{165}$	$z \geq 0, k \leq 37$ Cost $\geq 2^{161}$	$z \geq 50, k \leq 3$ Cost $\geq 2^{171}$	$z \geq 48, k \leq 31$ Cost $\geq 2^{165}$
	-1024	$z \geq 0, k \leq 33$ Cost $\geq 2^{229}$	$z \geq 26, k \leq 3$ Cost $\geq 2^{268}$	$z \geq 23, k \leq 11$ Cost $\geq 2^{263}$	$z \geq 0, k \leq 33$ Cost $\geq 2^{255}$	$z \geq 52, k \leq 3$ Cost $\geq 2^{269}$	$z \geq 50, k \leq 11$ Cost $\geq 2^{262}$
$\infty_{k=0}$	-512	$z \geq 22, k = 0$ Cost $\geq 2^{89}$	$z \geq 22, k = 0$ Cost $\geq 2^{97}$	$z \geq 22, k = 0$ Cost $\geq 2^{89}$	$z \geq 46, k = 0$ Cost $\geq 2^{90}$	$z \geq 46, k = 0$ Cost $\geq 2^{98}$	$z \geq 46, k = 0$ Cost $\geq 2^{90}$
	-768	$z \geq 22, k = 0$ Cost $\geq 2^{166}$	$z \geq 22, k = 0$ Cost $\geq 2^{171}$	$z \geq 22, k = 0$ Cost $\geq 2^{166}$	$z \geq 48, k = 0$ Cost $\geq 2^{165}$	$z \geq 48, k = 0$ Cost $\geq 2^{171}$	$z \geq 48, k = 0$ Cost $\geq 2^{165}$
	-1024	$z \geq 23, k = 0$ Cost $\geq 2^{263}$	$z \geq 23, k = 0$ Cost $\geq 2^{268}$	$z \geq 23, k = 0$ Cost $\geq 2^{263}$	$z \geq 50, k = 0$ Cost $\geq 2^{262}$	$z \geq 50, k = 0$ Cost $\geq 2^{268}$	$z \geq 50, k = 0$ Cost $\geq 2^{262}$

TABLE B.4: Summary of the values for the Jensen's gap 2^z at crossover points of our combined classical-quantum enumeration attacks against Kyber and the canonical 128, 192, 256 bit security respectively. We remark that exact crossovers happen at fractional values of z. In this table we round down threshold values of z. MAXDEPTH is abbreviated to MD. Cost is as in Table 7.6. The results are estimated from the bounds $\mathcal{C} = 2$, $\varepsilon = 20$, $b = 1/64$.

less likely to be feasible — more likely to be feasible

Crossover points when comparing quasi-square-root against $\log \mathbb{E}[\text{Quantum GCOST}]$ (cf. Equation (7.7)) with …

		…$\mathcal{W}$ as in Section 7.2.1			…$\mathcal{W}$ as in Section 7.2.2		
MD	Kyber	LB/UB	UB/UB	LB/LB	LB/UB	UB/UB	LB/LB
2^{40}	-512	$z \geq 0, k \leq 25$ Cost $\geq 2^{90}$	$z \geq 29, k \leq 28$ Cost $\geq 2^{126}$	$z \geq 20, k \leq 92$ Cost $\geq 2^{127}$	$z \geq 0, k \leq 39$ Cost $\geq 2^{126}$	$z \geq 44, k \leq 27$ Cost $\geq 2^{127}$	$z \geq 36, k \leq 96$ Cost $\geq 2^{126}$
	-768	$z \geq 10, k \leq 81$ Cost $\geq 2^{191}$	$z > 64$	$z \geq 64, k \leq 114$ Cost $\geq 2^{191}$	$z \geq 26, k \leq 69$ Cost $\geq 2^{191}$	$z > 64$	$z > 64$
	-1024	$z \geq 59, k \leq 105$ Cost $\geq 2^{254}$	$z > 64$	$z > 64$	$z > 64$	$z > 64$	$z > 64$
2^{64}	-512	$z \geq 0, k \leq 26$ Cost $\geq 2^{72}$	$z \geq 17, k \leq 11$ Cost $\geq 2^{126}$	$z \geq 8, k \leq 89$ Cost $\geq 2^{127}$	$z \geq 0, k \leq 24$ Cost $\geq 2^{98}$	$z \geq 33, k \leq 12$ Cost $\geq 2^{126}$	$z \geq 24, k \leq 69$ Cost $\geq 2^{127}$
	-768	$z \geq 0, k \leq 64$ Cost $\geq 2^{186}$	$z \geq 58, k \leq 37$ Cost $\geq 2^{191}$	$z \geq 52, k \leq 106$ Cost $\geq 2^{191}$	$z \geq 14, k \leq 67$ Cost $\geq 2^{191}$	$z > 64$	$z > 64$
	-1024	$z \geq 47, k \leq 100$ Cost $\geq 2^{254}$	$z > 64$	$z > 64$	$z \geq 63, k \leq 100$ Cost $\geq 2^{255}$	$z > 64$	$z > 64$
2^{96}	-512	$z \geq 0, k \leq 26$ Cost $\geq 2^{72}$	$z \geq 1, k \leq 4$ Cost $\geq 2^{126}$	$z \geq 0, k \leq 47$ Cost $\geq 2^{111}$	$z \geq 0, k \leq 26$ Cost $\geq 2^{97}$	$z \geq 17, k \leq 1$ Cost $\geq 2^{127}$	$z \geq 8, k \leq 40$ Cost $\geq 2^{127}$
	-768	$z \geq 0, k \leq 61$ Cost $\geq 2^{154}$	$z \geq 42, k \leq 26$ Cost $\geq 2^{191}$	$z \geq 36, k \leq 77$ Cost $\geq 2^{191}$	$z \geq 0, k \leq 67$ Cost $\geq 2^{187}$	$z \geq 59, k \leq 12$ Cost $\geq 2^{190}$	$z \geq 53, k \leq 70$ Cost $\geq 2^{190}$
	-1024	$z \geq 31, k \leq 100$ Cost $\geq 2^{254}$	$z > 64$	$z > 64$	$z \geq 48, k \leq 91$ Cost $\geq 2^{254}$	$z > 64$	$z > 64$
∞	-512	$z \geq 0, k \leq 26$ Cost $\geq 2^{72}$	$z \geq 0, k \leq 1$ Cost $\geq 2^{119}$	$z \geq 0, k \leq 40$ Cost $\geq 2^{111}$	$z \geq 0, k \leq 26$ Cost $\geq 2^{97}$	$z \geq 17, k \leq 1$ Cost $\geq 2^{127}$	$z \geq 8, k \leq 40$ Cost $\geq 2^{127}$
	-768	$z \geq 0, k \leq 37$ Cost $\geq 2^{136}$	$z \geq 4, k \leq 3$ Cost $\geq 2^{191}$	$z \geq 0, k \leq 31$ Cost $\geq 2^{187}$	$z \geq 0, k \leq 37$ Cost $\geq 2^{161}$	$z \geq 30, k \leq 3$ Cost $\geq 2^{191}$	$z \geq 22, k \leq 31$ Cost $\geq 2^{191}$
	-1024	$z \geq 0, k \leq 33$ Cost $\geq 2^{229}$	$z \geq 39, k \leq 3$ Cost $\geq 2^{255}$	$z \geq 31, k \leq 11$ Cost $\geq 2^{255}$	$z \geq 0, k \leq 33$ Cost $\geq 2^{255}$	$z > 64$	$z \geq 57, k \leq 11$ Cost $\geq 2^{255}$
$\infty_{k=0}$	-512	$z \geq 0, k = 0$ Cost $\geq 2^{111}$	$z \geq 0, k = 0$ Cost $\geq 2^{119}$	$z \geq 0, k = 0$ Cost $\geq 2^{111}$	$z \geq 9, k = 0$ Cost $\geq 2^{127}$	$z \geq 17, k = 0$ Cost $\geq 2^{127}$	$z \geq 9, k = 0$ Cost $\geq 2^{127}$
	-768	$z \geq 0, k = 0$ Cost $\geq 2^{188}$	$z \geq 2, k = 0$ Cost $\geq 2^{191}$	$z \geq 0, k = 0$ Cost $\geq 2^{188}$	$z \geq 22, k = 0$ Cost $\geq 2^{191}$	$z \geq 28, k = 0$ Cost $\geq 2^{191}$	$z \geq 22, k = 0$ Cost $\geq 2^{191}$
	-1024	$z \geq 31, k = 0$ Cost $\geq 2^{255}$	$z \geq 36, k = 0$ Cost $\geq 2^{255}$	$z \geq 31, k = 0$ Cost $\geq 2^{255}$	$z \geq 57, k = 0$ Cost $\geq 2^{255}$	$z \geq 63, k = 0$ Cost $\geq 2^{255}$	$z \geq 57, k = 0$ Cost $\geq 2^{255}$

Colophon

This thesis was typeset using LaTeX and the `memoir` documentclass. The template[1] is based on Friedrich Wiemer's thesis[2], which itself is based Aaron Turon's thesis [3], itself again a mixture of `classicthesis`[4] by André Miede and `tufte-latex`[5], based on Edward Tufte's *Beautiful Evidence*.

The bibliography was processed by Biblatex. The body text is set 10/14pt (long primer) on a 26pc measure. The margin text is set 8/9pt (brevier) on a 12pc measure. Matthew Carter's Charter acts as both the text and display typeface. Monospaced text uses Jim Lyles's `Bitstream Vera Mono` ("Bera Mono").

The thesis was written using *Sublime Text*[6] with the *LatexTools* Plugin, many of the graphics were build using *Ipe*[7]. The writing process was supported by the *write-good* linter[8] with the corresponding Sublime Text plugin[9]. Additionally, *Thesaurus*[10] and *DeepL*[11] were used to improve the wording of individual paragraphs.

[1] https://github.com/mtiepelt/dissertation

[2] https://github.com/pfasante/phd_thesis/tree/master

[3] https://people.mpi-sws.org/~turon/turon-thesis.pdf

[4] https://bitbucket.org/amiede/classicthesis/

[5] https://github.com/Tufte-LaTeX/tufte-latex

[6] https://www.sublimetext.com/

[7] https://ipe.otfried.org/

[8] https://www.npmjs.com/package/write-good

[9] https://github.com/btford/write-good

[10] https://www.thesaurus.com/

[11] https://www.deepl.com/de/write