TU WIEN Informatics

Post-Quantum Cryptography in OpenPGP

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Technische Informatik

eingereicht von

Aron Wussler, BSC
Matrikelnummer 11717702

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Univ. Prof. Matteo Maffei
Mitwirkung: Dipl.-Ing Erkan Tairi

Wien, 29. September 2023

Aron Wussler | Matteo Maffei

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

TU WIEN Informatics

Post-Quantum Cryptography in OpenPGP

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Computer Engineering

by

Aron Wussler, BSC
Registration Number 11717702

to the Faculty of Informatics

at the TU Wien

Advisor: Univ. Prof. Matteo Maffei
Assistance: Dipl.-Ing Erkan Tairi

Vienna, 29th September, 2023

Aron Wussler   Matteo Maffei

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

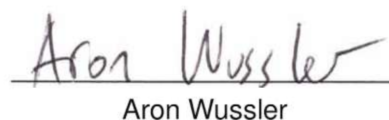
Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Erklärung zur Verfassung der Arbeit

Aron Wussler, BSC

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 29. September 2023



Aron Wussler

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Danksagung

Zuerst möchte ich mich bei allen, die an dem Projekt beteiligt waren und aktiv dazu beigetragen haben, bedanken: Dr. Falko Strenzke, Dr. Stavros Kousidis, Dr. Stephan Ehlen, Johannes Roth, Prof. Andreas Hülsing, Carl-Daniel Hailfinger und Dr. Evangelos Karatsiolis. Danke für die konstruktive Zusammenarbeit und die unzähligen Stunden, die wir gemeinsam bei der Diskussion des Projekts verbracht haben.

Meinen Kollegen möchte ich meinen aufrichtigen Dank aussprechen: Daniel Huigens für seine Unterstützung des Projekts und sein umfangreiches Wissen über das OpenPGP-Protokoll, Marin Thiercelin für seine Unterstützung bei der mobilen Implementierung von go-crypto und Marco Martinoli für das Teilen seines Wissens über Post-Quantum-Kryptographie.

Ich möchte mich auch bei Erkan Tairi und Prof. Matteo Maffei für die Anleitung und die angebotene Betreuung bedanken - ohne sie wäre diese Masterarbeit nicht möglich gewesen.

Mein Dank geht auch an die Mitglieder der Community für ihr Feedback zu allen offenen Fragen, sei es bei der IETF oder bei den OpenPGP-Gipfeln, besonders an: Daniel Kahn Gillmor, Justus Winter, Mike Ounsworth, Neal Walfield und Stephen Farrell.

Nicht zuletzt möchte ich meinen Eltern und meiner Lebensgefährtin danken, die ein paar Jahre Post-Quantum-Jargon ertragen mussten und wahrscheinlich noch viele weitere vor sich haben.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Acknowledgements

First of all, I would like to thank all those who participated and actively contributed to the project, Dr. Falko Strenzke, Dr. Stavros Kousidis, Dr. Stephan Ehlen, Johannes Roth, Prof. Andreas Hülsing, Carl-Daniel Hailfinger, and Dr. Evangelos Karatsiolis. Thank you for the constructive collaboration and the countless hours spent together discussing the project.

I wish to express the sincere gratitude to my colleagues: Daniel Huigens for his support of the project and vast knowledge of the OpenPGP protocol, Marin Thiercelin for his support with go-crypto's mobile implementation, and Marco Martinoli for sharing his knowledge regarding Post-Quantum cryptography.

I would also like to thank Erkan Tairi and Prof. Matteo Maffei, for the guidance and tutoring offered - without them this master thesis would not have been possible.

My gratitude goes also to the community members for providing feedback on all the open questions, either at the IETF or at the OpenPGP summits, including, but not limited to: Daniel Kahn Gillmor, Justus Winter, Mike Ounsworth, Neal Walfield, and Stephen Farrell.

Last but not least, I would like to thank my parents and my partner, who had to endure a couple of years of Post-Quantum jargon, and probably have many more to come.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Kurzfassung

Angesichts der neuesten Fortschritte im Quantencomputing [AAB+19] und des Shor's Algorithmus [Sho94] zielt dieses Projekt darauf ab, Post-Quantum (PQ) Kryptographie für die E-Mail- und Dateiverschlüsselung sowie die Softwareverteilung im OpenPGP-Protokoll zu integrieren.

14 neue Algorithmusbezeichner werden dem Protokoll hinzugefügt, die verschiedene Sicherheitsstufen von CRYSTALS-Dilithium und CRYSTALS-Kyber hybrid mit elliptischen Kurven und zwei eigenständige Varianten von SPHINCS$^+$ implementieren. Die Wahl und Konstruktion der Algorithmen werden ausführlich diskutiert, unter Berücksichtigung von Sicherheitsfaktoren, Benutzererfahrung und den Interessen verschiedener Stakeholder.

Das Projekt entwickelt auch eine Implementierung in Golang, welche die bestehende go-crypto Programmbibliothek erweitert, um sicherzustellen, dass der vorgeschlagene Standard problemlos implementiert werden kann und um die tatsächliche Leistung auf realen Desktop- und Mobilgeräten zu messen. Die Leistungsdaten werden dann sorgfältig analysiert, um Erkenntnisse über die zu erwartenden Unterschiede im Benutzererlebnis und die notwendigen Änderungen an den OpenPGP-Anwendungen zu erhalten.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Bibliothek, Your knowledge hub
TU WIEN
Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Abstract

Given the recent advancements in quantum computing [AAB$^+$19] and Shor's algorithm [Sho94], this project aims at bringing PQ cryptography for e-mail and file encryption as well as software distribution to the OpenPGP protocol.

14 new algorithm identifiers are added to the protocol, implementing different security levels of CRYSTALS-Dilithium and CRYSTALS-Kyber hybrid with elliptic curves, and two standalone variants of SPHINCS$^+$. An extensive discussion is provided on the algorithm choice and construction, considering security factors, user experience, and the interest of various stakeholders.

The project also develops an implementation in Golang expanding the existing go-crypto library, to ensure that the proposed standard can be cleanly implemented, and to measure the actual performance on real desktop and mobile devices. The performance data is then carefully analyzed to provide insights on the expected differences in user experience and the necessary changes to the OpenPGP applications.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

TU WIEN Bibliothek Your knowledge hub
Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Contents

Kurzfassung **xi**

Abstract **xiii**

Contents **xv**

1 Introduction **1**
1.1 Motivation . . . 2

2 Preliminaries **5**
2.1 OpenPGP . . . 5
2.2 Post-Quantum Schemes . . . 15
2.3 Key Derivation and Combination . . . 21

3 Protocol Design Changes **29**
3.1 Composite vs Composable . . . 29
3.2 Key Encapsulation Mechanisms . . . 34
3.3 Signature Schemes . . . 37
3.4 Artifact Encoding . . . 40
3.5 Version Binding . . . 40
3.6 Proposed Migration Strategies . . . 41

4 Implementation **43**
4.1 Dependencies . . . 43
4.2 Changes to the Library . . . 44

5 Results and Performance Analysis **51**
5.1 Artifact Size . . . 52
5.2 Desktop Performance Analysis Methodology . . . 54
5.3 Mobile Performance Analysis Methodology . . . 57
5.4 Key Generation . . . 57
5.5 Key Parsing . . . 61
5.6 Signature Generation . . . 64
5.7 Signature Verification . . . 67

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

5.8 Encryption 67
5.9 Decryption 71

6 Conclusion **73**
6.1 Results 74
6.2 Directions for Future Research 74

A Integral RFC Text **77**

List of Figures **117**

List of Tables **119**

Acronyms **121**

Bibliography **125**

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

CHAPTER 1

Introduction

This project aims at introducing Post-Quantum (PQ) cryptography for e-mail and file encryption, as well as software signing in the OpenPGP protocol. In the last years there has been a push among internet protocols to deploy asymmetric cryptography that can resists attacks with quantum computers. Other internet protocols, such as Transport Layer Security (TLS) [KSL+19, KV, ELPa] or Secure Shell Protocol (SSH) [ELPb, SKD20] have been experimenting with the PQ schemes to determine any performance drawback or issue with the larger artifacts.

This project, sponsored from Proton AG, starts at the beginning of 2021 with a first proposal to evaluate the feasibility of introducing PQ algorithms into OpenPGP and gather consensus in the community. At the same time, around June 2021, the German Federal Office for Information Security (BSI) has also started investing resources in this field of research, contracting the firm MTG AG to develop a standard and implement it into libgcrypt (the library behind GnuPG) and Botan. This plan was explained in detail from Dr. Falko Strenzke, on behalf of MTG AG, at Internet Engineering Task Force (IETF) 113th meeting, in Vienna. At this venue, a merger of the two projects was proposed, in order to come up with a common standardization plan led by the community.

The criteria chosen for the development of the standard was to use already standardized algorithm, mostly relying on National Institute of Standards and Technology (NIST) and the Crypto Forum Research Group (CFRG), to provide confidentiality and integrity. Given these cryptographic primitives, the project consists in developing a safe scheme, with reasonable and coherent security guarantees, considering the performance and regulatory compliance trade-offs. The main challenges consisted in choosing how to build the hybrid schemes, how many schemes to propose, how to combine the hybrid secrets securely, and how to handle the significantly larger artifacts.

The standardization proposal is based on the latest draft of the OpenPGP message format [WHWY23], and branches off to define new algorithms and the required protocol

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

changes.

Furthermore, to investigate the feasibility of the project, this thesis focuses on the implementation in the go-crypto Golang library, with the intention of comparing results and interoperating with MTG's libgcrypt implementation.

1.1 Motivation

The interest for post-quantum encryption gained particular traction in 1994, when Peter Shor found an algorithm to factor large numbers in polynomial time given a sufficiently large quantum computer [Sho94]. This well-known algorithm is nowadays believed to be useful in cracking the most commonly deployed algorithms for key exchange and asymmetric encryption based on the discrete logarithm problem and the factoring problem, like Rivest-Shamir-Adleman (RSA), Diffie-Hellman (DH), Digital Signature Algorithm (DSA), Elliptic Curve Diffie-Hellman (ECDH), Elliptic Curve Digital Signature Algorithm (ECDSA), Edwards-curve Digital Signature Algorithm (EdDSA), and ElGamal [BL17].

By then some non-vulnerable algorithms already existed, starting with R. J. McEliece's paper "A public-key cryptosystem based on algebraic coding theory" [McE78] which uses Error-Correcting Codes, in particular Goppa codes. The following year, the first post-quantum digital signature scheme was published: Lamport's idea was based on hash functions [Lam79]. More recent schemes are based on the Shortest Vector Problem (SVP) in multi-dimensional lattices [GGH96, Ajt96] or multivariate quadratic equations [MI88].

Even though to the best of our knowledge, a quantum computer capable of cracking the state of the art does not yet exist, we can refer to Mosca's theorem [EHH+20], to understand why this is already relevant. Let x be the number of years that the data to be protected must remain secured, y be the number of years needed to convert the corresponding system to quantum computer resistant cryptography, and z be the number of years it will take for quantum computers to exist that threaten the cryptography currently in use. Standardization needs to happen before $x + y > z$.

Considering the recent advancements in quantum computing [AAB+19] the BSI believes that quantum computers may already become a threat already in the early 2030s [BT120]. In the US, NIST has started a competition in 2017 to standardize post-quantum algorithms, that has now completed the third round [MAA+22].

Standardization is a lengthy and complex process, and given that the encrypted data is supposed to stay confidential for the years to come, we need to ensure that the process is carried out early enough. Request For Comments (RFC) 7258 [FT14] states that pervasive monitoring is an attack that should be mitigated in the design of IETF protocols (such as OpenPGP) and defines some guidelines for its mitigation. This attack, also known as "store now and decrypt later", has to be addressed providing the community with PQ tools to use as soon as possible. Signatures, on the other hand, are yet not subject to this vulnerability, since they can be re-issued without losing integrity if necessary.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Other internet protocols such as TLS [KSL+19, KV, ELPa] or SSH [ELPb, SKD20] have already started standardizing PQ algorithms, while OpenPGP is lagging behind: a few proposals were raised at the 2018 OpenPGP summit, but this did not lead to any further research attempt.

In the next chapter, the state of the art of post-quantum encryption is going to be discussed, giving particular attention to the OpenPGP context and requirements. The selected algorithms will then be justified considering the advantages and challenges of implementing PQ algorithms in an asynchronous (and potentially offline) protocol like OpenPGP.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

CHAPTER 2

Preliminaries

2.1 OpenPGP

This protocol was originally created from Philip R. Zimmerman as Pretty Good Privacy (PGP) in 1991, to encrypt e-mail messages. In 1997, due to some patent and interoperability issues, PGP Inc. decided to open the standard, called it OpenPGP and defined it in RFC 2440 [FTDC98]. This document standardized the original OpenPGP Message Format, that has since been obsoleted by RFC 4880 [FDC+07]. Many additions have been added to this protocol since its publication in 2007, for instance the NIST elliptic curves in RFC 6637 [Jiv12], or Ed25519 [Koc16].

All these different specifications that got adopted over the years are being consolidated in a single document [WHWY23], informally named “crypto-refresh”. It defines new versions for keys and packets with significant security improvements, obsoletes old algorithms, consolidates the standardization of elliptic curves, and aims at better interoperability among implementations.

All the development of PQ algorithms has been based on this document: PQ schemes are designed to work only with the latest version 6 keys. The rationale is that users will need to update in order to benefit from any new algorithm, therefore we can benefit from security advantages without adding any further upgrade burden.

This latest specification has grown considerably, mostly because the use cases of OpenPGP expanded over the years, but also due to the legacy burden. The OpenPGP protocol has different requirements from those of other protocols where PQ algorithms have been already tested out, such as TLS or SSH. OpenPGP is generally asynchronous: keys are usually fetched once from a remote server, and then used when necessary to produce ciphertexts offline. Clearly, this introduces some challenges and opportunities that need to be considered when designing an implementation, and the designs for TLS or SSH can not be blindly reused.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

2.1.1 Use Cases

In this section the use cases of OpenPGP are going to be listed, in order to understand the protocol constraints and later explain its current design. The changes proposed in this project introduced in chapter 3 are then based off the current structure and use cases.

Legacy Burden

The most important problem with OpenPGP is deprecation: it is significantly harder to deprecate algorithms compared to real-time key exchange protocols. Once keys using a specific algorithm have been issued, it is usually a long process to remove the algorithm from the implementation. In fact, in order to send a message, the sender needs to support the encryption schemes required by the recipient, if not no communication is possible. Furthermore, when decrypting old ciphertexts we face another challenge: in order to decrypt them we still need to support the scheme used to encrypt. Old data would have to be re-encrypted in order to drop support for deprecated algorithms.

Compared to TLS, where defining an experimental Key Encapsulation Mechanism (KEM) is simple, and the real-time algorithm negotiation will exclude it once deprecated, in OpenPGP this is not possible, and including a scheme means committing for a longer time to it.

Real-Time Requirements

Unlike TLS or SSH, OpenPGP has no real-time requirements. This means that a reasonably computationally expensive key exchange or signature scheme can be implemented without heavy impact on the user experience. This also allows larger keys, since they are rarely fetched.

However, this may change as adoption of Web Key Directory (WKD) increases: WKD is a recent development in the standard that provides an easier infrastructure to fetch public keys in real time [Koc22]. This would increase the versatility of short-lived keys, but gives more importance to reducing public key size.

Signing-capable OpenPGP keys can also be used in authentication mode, for instance for SSH authentication. This use is rarely seen in the wild, and therefore not a primary focus of this project.

Long-Lived Keys

Since 2017 TLS certificate lifetime is capped to 825 days [n.A17]. This maximum duration provides a better key rotation, setting the prerequisites to deprecate old hash algorithms or enforce new minimum security parameters. This does not happen in OpenPGP, where it is common to generate keys without expiration or encounter keys that are several years old.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

This aspect, joint with the legacy burden, makes it very important to choose algorithms that provide a higher degree of safety, and offer options that exceed today's computational expectations.

Secure Enclaves

Another common use for OpenPGP keys is in Hardware Security Module (HSM) chips or smart cards. All the ciphers in the NIST competition are required to be possible to be implemented on HSMs, even though for many of them the hardware acceleration is still in research or development [HMOR21][1]. By picking standardized and widely adopted algorithms, there is a greater chance that optimized designs for these might be readily available in the near future.

Governments and Policy Restrictions

OpenPGP is widely used from several governmental agencies, and therefore its standardization and use is vetted from the relevant agencies and standardization bodies. For instance, many entities require Federal Information Processing Standard (FIPS) compliant algorithms, such as the NIST curves, while other governments push for other curves, for instance the BSI [EHH+20] with its requirement for Brainpool [ML10] curves.

Given the many interests and parties, the right balance between implementation burden and compliance must be found. Considerations from CFRG [Hof20], the National Security Agency (NSA) [Age22], and the BSI [fIS20] were given significance to ensure continued use of the OpenPGP standard and adoption of this project.

Key Distribution

OpenPGP does not have a unified key distribution system, and relies on off-band methods. Some of the most common methods are:

- WKD, a domain-bound key distribution system which provides domain authentication via Hypertext Transfer Protocol Secure (HTTPS).
- Public Key Server (PKS), a legacy key distribution system where public keys can be uploaded without any sort of verification.
- Hagrid, an evolution of PKS that enforces address ownership verification and transmits keys over HTTPS.
- Autocrypt, an email header that signals OpenPGP support and contains the encoded public key.

[1] In this paper, table 1 presents a summary of the current state-of-the-art of hardware designs of NIST post-quantum candidates, implemented on Field Programmable Gate Array (FPGA) circuits

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

- E-mail attachment, explicitly attaching the public key in a Multipurpose Internet Mail Extensions (MIME) e-mail.

- Other manual systems, such as uploading to a website or physical file sharing.

Most of the keys are distributed over HTTPS, that does not put a strict cap on the public key size. Nevertheless, some schemes' performance and viability might be hampered if the public key size is larger than some megabytes, in particular the header-based Autocrypt and e-mail attachments.

2.1.2 Protocol Design

Considered the use cases and requirements of the protocol, we now flesh out the current protocol design. The protocol described here includes the changes included in the latest draft specification [WHWY23].

We will start with a high-level description of the two most common operations of this protocol, encryption to provide confidentiality and signature to provide authenticity.

Encryption In order to encrypt a message, this is first encoded, optionally compressed, and then symmetrically encrypted using a randomly-generated session key. This session key is then encrypted asymmetrically to the various recipients public keys' or symmetrically to a password, to be communicated off-band.

Signature A signed message can be encrypted or in cleartext. In the first case, the message is encoded, then a signature is computed over a hash of the encoded data. Some metadata about the signature is prepended to the message, and the signature data itself is appended. The signed message is then encrypted. In the second case, the message is first normalized, then signed, and finally encoded with the signature in a special armoring so that it can be read by humans without decoding, and at the same time preserve its format to reliably allow verification.

OpenPGP is not only for messages or files, and can also be used for a variety of other purposes, such as passwordless authentication, software distribution, identity management, and so on. All these functions are achieved by the *packet* data encoding format.

Packets and Sub–packets

The OpenPGP wire format encodes data in packets, some of which contain sub-packets. All packets are prefixed with a `tag`, that identifies the type, and a `length`, that allows splitting and parsing. Tags can be critical or non-critical. If a critical tag is not understood from the parsing implementation the message is classified as unreadable. Here we analyze the inner structure of the relevant packet types specified so far.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Tag 1: Public-Key Encrypted Session Key (PKESK) Packet This packet contains a session key, encrypted to a recipient's public key. It is one of the most common packets, enabling Public-Key Encryption (PKE).

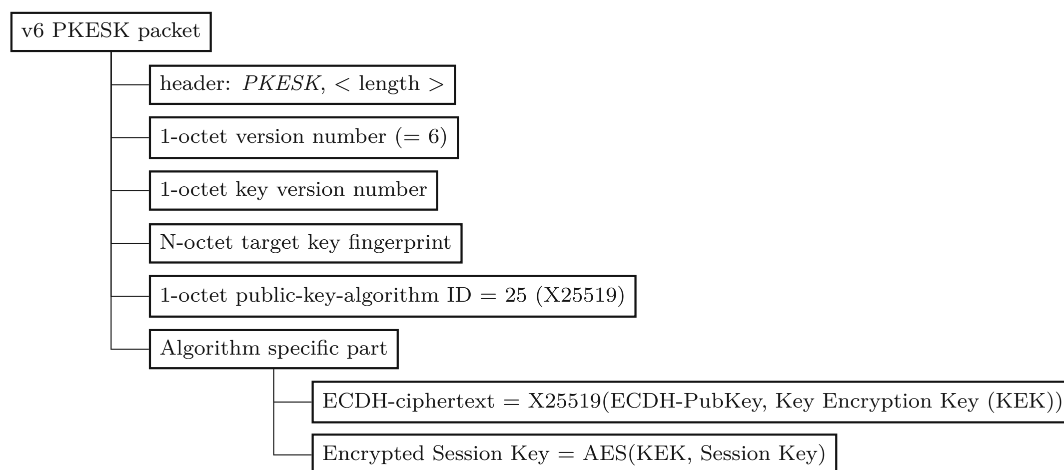


Figure 2.1: Version 6 of an X25519 PKESK packet structure.

The packet structure, illustrated in fig. 2.1, shows the data encoding for a version 6 PKESK for the X25519 algorithm. It contains a version to allow parsing, a fingerprint to identify the target key, and the public key algorithm, directly followed by the algorithm-specific data. In this part, each algorithm can define a way to encode the ciphertext and/or a wrapped session key.

Tag 2: Signature Packet This packet contains the complete data for a signature: with the matching public key and signed data this allows verification.

An example of a version 6 signature packet for Ed25519 can be seen in fig. 2.2. It contains significantly more metadata than a PKESK, including a signature type to encode the intended use and an arbitrary amount of subpackets to encode signature properties.

Signatures are in fact used in many different parts of OpenPGP, not just to sign messages but also to bind identities, certify or revoke keys, and so on. The type provides the necessary domain separation to identify the purpose of a signature.

Subpackets instead provide some extra information, such as creation time, lifetime, issuer, or other preferences. Two types of subpackets are to be distinguished: hashed and unhashed. The former is serialized into the signed hash, and therefore certified from the issuer, while the latter is for advisory information, uncertified. Most subpackets are required to be in the hashed portion, with a few exceptions, notably the issuer metadata.

OpenPGP uses a hash-then-sign paradigm on the protocol level, in order to implement easier data streaming. The data followed by the hashed subpackets is first hashed, then a signature is produced over the hash itself.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

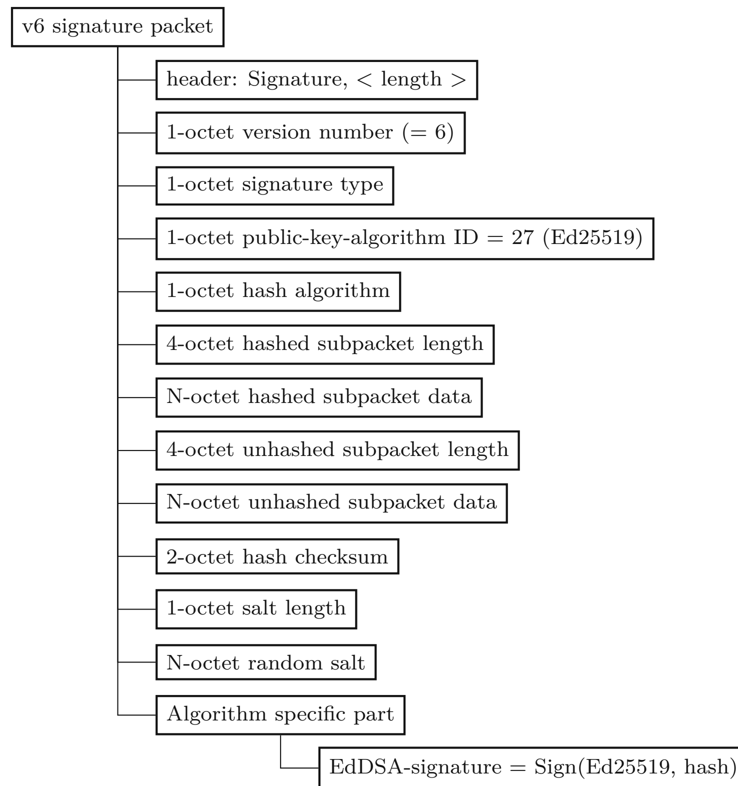


Figure 2.2: Version 6 Ed25519 signature packet structure.

Furthermore, version 6 signatures include an unpredictable salt to be prepended to the signed data as an additional security feature. It ensures that weak-collision resistance is sufficient for the hash function used in the data hashing [LP20]. An adversary given access to a signing oracle, may not query for a specific string, since the hash will be randomly initialized with the salt. Therefore, in order to create a valid signature, for the adversary is not sufficient to find

$$H(x) = H(y) \quad \text{with } x \neq y$$

but it is required to find

$$H(x) = H(y) \quad \text{given } x.$$

In the scope of the project, the latest specification was updated to bind the salt size to the hash function expected security level, i.e. 16 bytes for 256-bit hash functions, 24 bytes for 384-bit hash functions, and 32 bytes for 512-bit hash functions.

Tag 3: Symmetric-Key Encrypted Session Key (SKESK) Packet This packet encrypts a session key to a password, to symmetrically encrypt a message for a recipient not having a public key or preferring a password protected message.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

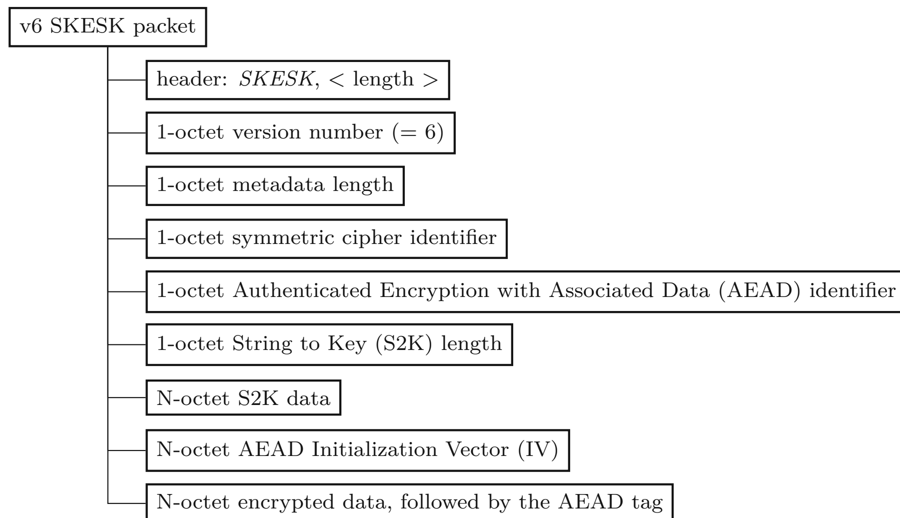


Figure 2.3: Version 6 SKESK packet structure.

The SKESK packet structure is illustrated in fig. 2.3. This packet derives a KEK from a password using an S2K, then uses it to wrap the session key using AEAD. The S2K contains a salt, and in the latest specification Argon2 [BDKJ21] may be used for key derivation.

This packet requires no changes to become PQ, as there is no known effective quantum-computing enabled attack against the supported symmetric algorithms or S2Ks.

Tag 4: One-Pass Signature (OPS) Packet The signature metadata is prepended to the message using this packet. Since OpenPGP uses a hash-and-sign paradigm, some metadata is necessary to instantiate the hashing function. It does not contain the signature values, therefore does not allow verification by itself.

In fig. 2.4 the OPS packet structure is illustrated. This packet contains the issuer key fingerprint, to detect if the verification key is available, as well as the hash algorithm used and the salt to be prefixed to the data. Any discrepancy between this packet and the signature packet appended to the signed data is required to result in an invalid signature. This packet does not contain any algorithm-specific data.

Tag 5: Secret-Key Packet The secret and public key material are wrapped by this packets. It is a strict superset of the public key packet. The secret key material can be encrypted, deriving the encryption key from a salted password, optionally using Argon2 [BDKJ21].

The packet structure is illustrated in fig. 2.5b. On top of the public key material, it contains the S2K parameters to derive the key from a password, the encryption parameters, and the AEAD encrypted algorithm-specific key material.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

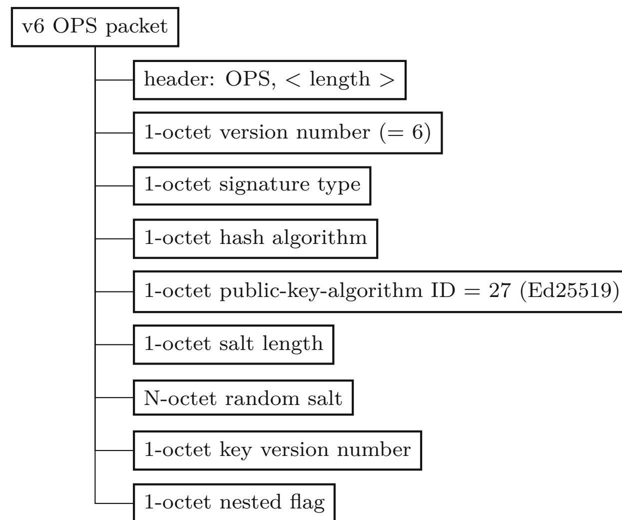
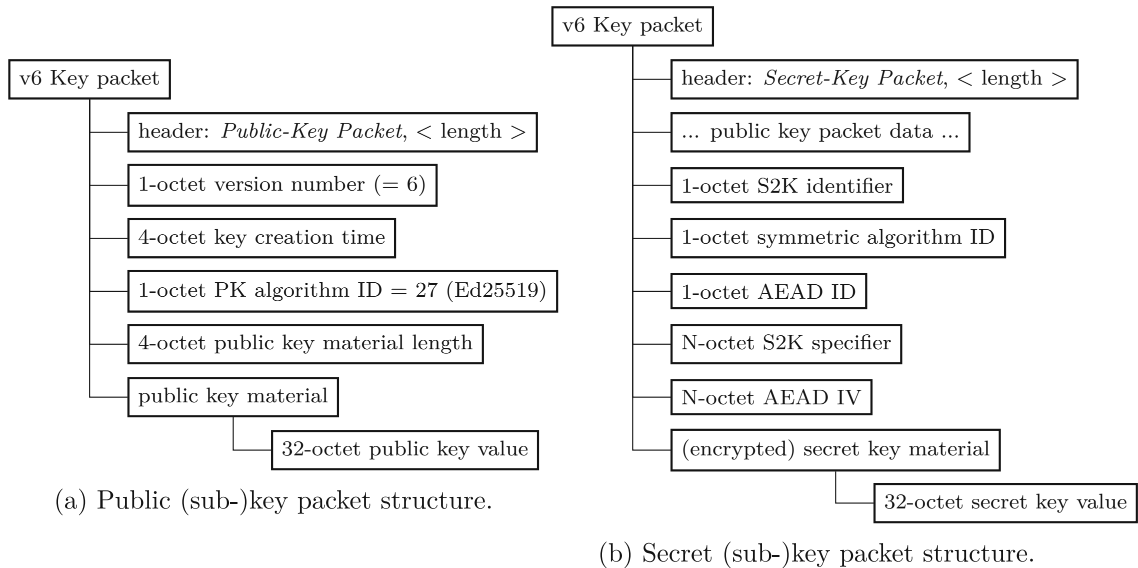


Figure 2.4: Version 6 OPS signature packet structure.

Tag 6: Public-Key Packet The public key material is encoded here. This packet is hashed to derive the key fingerprint, an univocal identifier of the key. This packet is used only for primary keys, therefore the key represented is required to be signing-capable to certify other subkeys.



(a) Public (sub-)key packet structure.

(b) Secret (sub-)key packet structure.

Figure 2.5: Version 6 public and secret key packets. The secret key is a superset of the public key, containing all of its parameters, plus the optionally encrypted algorithm-specific secret key material.

The structure of this packet is illustrated in fig. 2.5a. It is relatively simple, containing

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

the creation time, the public-key algorithm ID, and the algorithm-specific public key material.

Tag 7: Secret-Subkey Packet A private key can have attached an arbitrary number of sub-keys to perform authentication, signature, or decryption. They are certified by a signature made by the primary private key.

This packet has the same structure as a Secret-Key packet.

Tag 8: Compressed Data Packet This packet applies a compression algorithm to the contained data: it is used to compress plaintext before encryption. As other instances where compression is applied to plaintext, this may open to compression attacks, i.e. a way to exfiltrate information from the ciphertext observing the encrypted length.

This packet's structure is simply composed by one octet representing the algorithm used to compress the packet followed by the compressed data, which makes up the remainder of the packet.

Tag 9: Symmetrically Encrypted Data Packet This is a deprecated packet that was used to encrypt data without integrity. It enabled the EFAIL [PDM+18] attack and is now rejected by default from most implementations. In this attack, an adversary can change part of the ciphertext in a way that upon decryption trigger some actions into the mail client, possibly sending the content of the encrypted mail to the attacker. Given that there is no integrity protection it is not possible to prevent an adversary from altering the ciphertext without a failure on the decryption level.

Due to this vulnerabilities, generation of this packet is forbidden, and is not considered in the scope of this project. Any PQ implementation will be mandated not to encrypt data using this method.

Tag 11: Literal Data Packet The wrapper around plaintext to allow plaintext serialization inside encrypted messages.

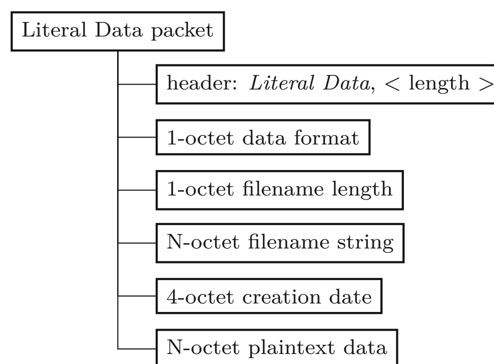


Figure 2.6: Literal data packet structure.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

The simple structure of the packet is represented in fig. 2.6. It contains the plaintext with some metadata, such as creation date or filename.

Tag 13: User ID Packet This packet contains an RFC2822 [Res01] formatted identity, and bound via a self-signature to the key.

Tag 14: Public-Subkey Packet This is the public equivalent of private subkeys. They can be attached to public keys via a binding signature.

This packet has the same structure as a Public-Key packet.

Tag 18: Symmetrically Encrypted Integrity Protected Data (SEIPD) This packet symmetrically encrypts data using the session key and an integrity protected encryption mode.

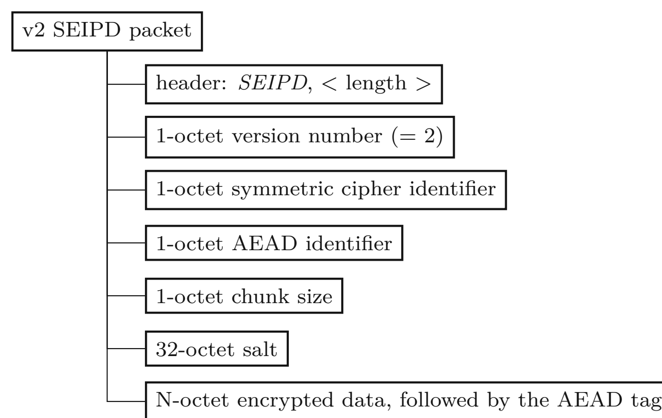


Figure 2.7: Version 2 SEIPD packet structure.

The packet version 2 structure is represented in fig. 2.7. It contains in cleartext the metadata used for encryption, such as the cipher suite, AEAD mode, block size, and a salt. All of these parameters are then fed into HMAC-based Key Derivation Function (HKDF) [KE10], using SHA-256 as underlying hash function, to derive the data encryption key and the IV.

Certificate Structure

An OpenPGP certificate is generally composed by the following series of packets:

- A certification-capable primary key packet (Tag 5 or 6) that provides a fingerprint, to identify the key itself.
- A direct key-signature (Tag 2) that contains the key algorithm preferences.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

- Zero or more User ID packets (Tag 13) to identify the user of the key, followed by a binding signature made from the primary key to ensure integrity.
- Zero or more sub-keys (Tag 7 or 14) used for KEM, signature, and authentication. These are also bound to the primary key through a signature.

Given this structure it is interesting to provide at least one very strong post-quantum signature scheme to ensure that primary keys can be long-lived, swapping only the sub-keys.

Furthermore this key structure can be used for a greedy approach to fight pervasive monitoring: we can attach PQ KEM subkeys to an existing traditional primary key, ensuring backwards compatibility while allowing already PQ implementations to secure the key exchange for the future.

Encrypted Message Structure

An OpenPGP signed and encrypted message is generally encoded with the following series of packets:

- One or more PKESK (Tag 1) or SKESK (Tag 3) encrypting the session key with a KEM or password.
- An SEIPD packet (Tag 18), using the session key to encrypt the following:
  - One or more OPS packets (Tag 4) containing the signature metadata.
  - A literal data packet (Tag 11), wrapping the user input.
  - A number of signature packets (Tag 2) corresponding to the the number of OPS packets signature packet, containing the different signatures' payloads.

  This payload may optionally be embedded in a compressed data packet (Tag 8).

This structure is designed to encrypt to multiple recipients: the same session key is wrapped for each one of them.

Furthermore, the message may contain multiple signatures, also from the same sender. This feature can be used to smoothen the transition, embedding both a traditional and a PQ signature for the same message.

2.2 Post-Quantum Schemes

Our review of the existing literature will be based on the results of the third round of the NIST PQ competition [MAA+22]. We will first present the selected candidates in detail, then elaborate on why they were preferred over other alternatives.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

2.2.1 Lattice-based Schemes

Lattices are discrete subgroups of n-dimensional real vector spaces with a zero element. Elements are vectors that fulfil the properties of addition and have an inverse. Given any two vectors there also exists a minimum distance, i.e. they can not be arbitrarily close.

This concept lays the foundation of the SVP problem: find the shortest vector given two arbitrary points of the lattice. This procedure is assumed to be hard for lattice in high enough dimensions. Most lattice-based PQ algorithms do not rely on the SVP directly, but rather are designed on the Learning With Errors (LWE) problem, that can be shown to be asymptotically as hard as a variant of the SVP [EHH+20].

The LWE problem consists in finding the vector $\mathbf{s}$ in a noisy linear system $\mathbf{b} = A\mathbf{s} + \mathbf{e}$ where $\mathbf{e}$ is a small vector in the lattice. In particular, given a vector $\mathbf{s} \in \mathbb{Z}_q^n$ and error distribution χ, define the LWE distribution $A_{\mathbf{s},\chi}$ over $\mathbb{Z}_q^n \times \mathbb{Z}_q$ by choosing $\mathbf{a} \in \mathbb{Z}_q^n$ uniformly at random, sampling $e \leftarrow \chi$ over $\mathbb{Z}$, and outputting the pair $(\mathbf{a}, b)$ where $b = \langle \mathbf{s}, \mathbf{a} \rangle + e \mod q$. We then define the two following problems:

- The Search-LWE$_{n,m,q,\mathcal{B},\chi}$ problem: let $\mathbf{s} \in \mathbb{Z}_q^n$ be chosen from some distribution $\mathcal{B}$. Given m samples $(\mathbf{a}_1, b_1), \ldots, (\mathbf{a}_m, b_m) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ drawn independently at random from the distribution $A_{\mathbf{s},\chi}$, find $\mathbf{s}$.
- The Decision-LWE$_{n,m,q,\mathcal{B},\chi}$ problem: let $\mathbf{s} \in \mathbb{Z}_q^n$ be chosen from some distribution $\mathcal{B}$. Without knowing $\mathbf{s}$, given m samples $(\mathbf{a}_1, b_1), \ldots, (\mathbf{a}_m, b_m) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$, distinguish between the following two cases:
    1. The samples are drawn independently from the distribution $A_{\mathbf{s},\chi}$.
    2. The samples are drawn independently from the uniform distribution on $\mathbb{Z}_q^n \times \mathbb{Z}_q$.

An example of encryption scheme based on this hardness problem is FrodoKEM, that was excluded from the 4th round of the NIST competition, given that a more performant lattice-based KEM has been selected for standardization. The performance is obtained by adding algebraic structure to the underlying lattice, for instance using the Module Learning With Errors (MLWE) problem. The hardness problem "find $\mathbf{s}$ given $\mathbf{b} = A\mathbf{s} + \mathbf{e}$ where $\mathbf{e}$ is a small vector in the lattice" is the same as in the LWE assumption, but $\mathbf{b}$, $\mathbf{e}$, and $\mathbf{s}$ are now vectors of polynomials. In particular, given a degree-n polynomial ring of the form $R = R_q = \mathbb{Z}_q[X]/(p(X))$, for some positive integer q and a polynomial $p(X)$, we define the following problem:

- The Decision-MLWE$_{R,m,k,q,\mathcal{B},\chi}$ problem: let $\mathbf{s} \in R_q^k$ be chosen from some distribution $\mathcal{B}$. Without knowing $\mathbf{s}$, given m samples $(\mathbf{a}_1, b_1), \ldots, (\mathbf{a}_m, b_m) \in R_q^k \times R_q$, distinguish between the following two cases:
    1. Every sample is drawn independently from the distribution $A_{R,\mathbf{s},\chi}$, analogue to the LWE distribution $A_{\mathbf{s},\chi}$, but over R_q.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

2. Every sample is drawn independently from the uniform distribution on $R_q^k \times R_q$.

Compared to the LWE, the MLWE increases efficiency and reduces key size, therefore researchers have been focused on in its security by looking for reductions for between average and worst-case [LS12]. The two main candidates for this project are CRYSTALS-Kyber and CRYSTALS-Dilithium, both MLWE schemes.

CRYSTALS-Kyber

Kyber [ABD+21] is a KEM based on the MLWE problem. Its PKE scheme uses elements in R_q and R_q^k where R is a cyclotomic power-of-2 ring $R := \mathbb{Z}[X]/(X^{256}+1)$, q is the prime 3329, and k is the module rank, set to 2, 3, or 4 depending on the security level. Elements are sampled from χ, a distribution of "short" polynomials of R_q.

To generate a key, a matrix of random polynomials $A \in R_q^{k\times k}$ is pseudorandomly generated from a uniformly random string. Then two secret vectors of polynomials $\mathbf{s}, \mathbf{e} \in R_q^k$ are sampled independently from χ coefficient-wise. The vector $\mathbf{s}$ is regarded as the secret key, and the vector $\mathbf{e}$ is called the error term. The MLWE public key is $(A, A\mathbf{s} + \mathbf{e})$.

To encrypt a message m two vectors of polynomials $\mathbf{r}, \mathbf{e}_1 \in R_q^k$ and a polynomial $e_2 \in R_q$ are sampled, with all coefficients of each polynomial chosen independently from χ. Then, the ciphertext c is formed as

$$c := (c_1, c_2) := \left(\mathbf{r}A + \mathbf{e}_1, \mathbf{r}\mathbf{b} + e_2 + \left\lceil \frac{q}{2} \right\rfloor \cdot m\right) \in R_q^k \times R_q$$

To decrypt a ciphertext c using the secret key $\mathbf{s}$ the intermediate value $v = c_2 - \mathbf{c}_1\mathbf{s}$ is computed, then each coefficient of the polynomial $v \mod 2$ is rounded to extract the encoded m.

Kyber offers good performance, comparable to elliptic curves, and artifact size in the 1 to 3 KB range for security level 3 and 5. NIST believes that MLWE is suitable for high-performance cryptosystems without sacrificing security, and the Kyber team provides an extensive security analysis with concrete estimates for the security parameters. It is therefore the only KEM candidate chosen for standardization, eventually giving it FIPS compliance and setting it up for wide adoption with solid and reviewed implementations.

Kyber has some known patent issues, but in November 2022 the intellectual property agreement was released [Cou22], stating that adopting the NIST standardized version of Kyber is going to be royalty-free. Given the legacy burden of OpenPGP, wide deployment is not considered before the proposed algorithms are fully standardized, therefore this scheme is suitable for implementation in the context of this project.

As recommended by the authors, it is proposed to be used in composite mode with well-established curves, ensuring that any flaw in the algorithm or implementation does not compromise data security.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

CRYSTALS-Dilithium

Dilithium [DKL+21] is a signature scheme based on the decisional MLWE assumption, under which the private key can not be recomputed from the public key. In particular, it uses the ring $R_q := \mathbb{Z}_q[X]/(X^{256}+1)$, where q is the prime number $2^{23} - 2^{13} + 1$. The public key is the sample $(A, \mathbf{t} := A\mathbf{s}_1 + \mathbf{s}_2)$, where A is a matrix over R_q and $\mathbf{s}_1$ and $\mathbf{s}_2$ are error vectors over R_q.

To sign a message, the scheme uses a "Fiat-Shamir with aborts" approach, where given a random $\mathbf{y}$, the signature consists of the high bits of vector $\mathbf{z} := \mathbf{y} + c\mathbf{s}_1$. Here, $c \in R_q$ is generated via the Fiat-Shamir procedure, $c = \text{hash}(m||A\mathbf{y})$ for a message m. If the challenge c does not have a specific distribution, then the vector $\mathbf{z}$ can reveal information about $\mathbf{s}_1$, therefore the procedure is aborted and repeated with a different random $\mathbf{y}$. The public key is then compressed.

This scheme ensures a high throughput, with performance close to elliptic curve schemes, and artifacts in the 1 to 5KB range for security level 3 and 5. Furthermore it has been chosen as main candidate from NIST, carrying the same advantages as Kyber. Finally it has been successfully implemented on FPGA circuits [RMJ+21].

For all these reasons, this scheme was adopted in the proposal in composite mode with well-established curves, similarly to Kyber.

2.2.2 Hash-based Schemes

This category consists of signature schemes based on the security assumptions of the underlying hashes. Leslie Lamport and Ralph Merkle started this field in 1979 [Lam79, Mer79], introducing the Merkle signatures. A drawback of these schemes is statekeeping. Being One-Time Signature (OTS) schemes, they require the signing end to keep track of which signature keys have already been used. If a signature key is reused, it can be compromised.

For hash-based schemes, stateful vs stateless represents a very important distinction. Stateful schemes are generally simpler, have compact and efficient signatures, and have already been standardized as Leighton-Micali Signatures (LMS) in RFC 8554 [MCF19] and as Extended Merkle Signature Scheme (XMSS) in RFC 8391 [HBG+18]. On the other hand, they require very precise statekeeping, and therefore need to ensure only one copy of the key is being used at a time. Their application is often limited to special hardware that does not allow key export, and is extremely impractical in the OpenPGP context, where there is no synchronized key management. We will therefore focus the analysis on stateless schemes, in particular SPHINCS$^+$, as they offer the versatility required for the OpenPGP protocol.

SPHINCS$^+$

SPHINCS$^+$ [ABB+22b] combines the use of one-time signatures, few-time signatures, Merkle trees, and hypertrees to generate a general-purpose scheme that does not require

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

statekeeping and allows up to securely generate 2^{64} signatures for each key pair. It is the only stateless hash-based candidate that is considered in the NIST standardization process, and the only hash-based scheme considered in the scope of the project.

In particular, the private key contains two uniformly sampled elements. First, the n-byte secret seed `SK.seed`, which is used to generate all the Winternitz One Time Signature (WOTS+) and Forest Of Random Subsets (FORS) private key elements. Second, an n-byte Pseudo-Random Function (PRF) key `SK.prf` which is used to deterministically generate a randomization value for the randomized message hash.

The public key also contains two elements. First, the Hyper-Tree (HT) public key, i.e. the root of the tree on the top layer. Second, an n-byte public seed value `PK.seed` which is sampled uniformly at random.

Generating a signature consists of the following steps:

1. A random value r is pseudorandomly generated from the message and `SK.prf`;
2. The value r is then used to compute an m-byte message digest which is split into a $\lfloor (k \log t + 7)/8 \rfloor$-byte partial message digest `md`, a $\lfloor (h - h/d + 7)/8 \rfloor$-byte tree index `idx_tree`, and a $\lfloor (h/d + 7)/8 \rfloor$-byte leaf index `idx_leaf`;
3. The values `md`, `idx_tree`, and `idx_leaf` are truncated to the necessary number of bits;
4. The partial message digest `md` is then signed with the `idx_leaf`-th FORS key pair of the `idx_tree`-th XMSS tree on the lowest HT layer;
5. The public key of the FORS key pair is then signed using HT.

A signature is therefore composed of the concatenation of r, the FORS signature, and the HT signature.

Signature verification consists of the following steps:

1. Recomputing message digest `md`, `idx_tree`, and `idx_leaf` using the same procedure as signature generation;
2. Computing a candidate FORS public key from `idx_tree`, `idx_leaf`, and the FORS signature;
3. Verifying the HT signature on the candidate FORS public key.

The scheme is rather complex, but its security proof relies exclusively on the security assumptions of the underlying hash: SHA-256, SHAKE-256, or Haraka. SPHINCS$^+$ offers also a wide selection of parameters for tradeoffs between security, signature generation speed, and signature size.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

As a comparison with Dilithium, to achieve the comparable security level, signing with `SPHINCS+-SHA-256-192f-simple` requires 65 million CPU cycles against 0.4 million for `Dilithium3`, to produce a signature over 10 times in size.

This scheme represents a great candidate for OpenPGP primary keys or long-lived signatures as it provides long-term security guarantees, at the expense of greater bandwidth usage and slower signature generation. In the context of the project only SHA-2 and SHAKE hash functions were considered. The former because of the performance, as they require less than half of the CPU cycles to produce a signature, the latter to be future-proof. Haraka was excluded because its security level is capped at 2, and it represents unnecessary implementation burden.

The scheme is included in the proposal as standalone, as the confidence in its proof and the security of the underlying hashes is very high.

2.2.3 Schemes Excluded from the Project

To prevent the specification from implementing too many algorithms and counter the legacy burden, after careful consideration we excluded Falcon, Classic McEliece, and BIKE from the project.

In particular, Falcon [FHK+20] is a signature scheme over NTRU lattices using fast-Fourier sampling. This procedure requires floating-point arithmetics, therefore making implementations complex, especially if constant-time operations are desired. In comparison to the other candidates, it features compact proofs, but given that bandwidth is not a critical requirement for OpenPGP, CRYSTALS-Dilithium appears as a better lattice-based candidate. To reduce the legacy burden we decided to limit the number of lattice-based candidates to one.

Classic McEliece [ABC+22], instead is a KEM based on the assumptions that binary Goppa codes it uses are indistinguishable from random linear codes, and that random linear codes can only be decoded with exponential effort due to the General Decoding Problem, also on quantum computers. The scheme has been subject to many years of scrutiny with little changes to the security parameters due to new attacks, therefore providing strong confidence in its security level. On the other hand, to achieve secure communication, Classic McEliece presents slow key generation and requires large public key size. While the former is not an issue in the OpenPGP context, the latter can be problematic with respect to some key distribution systems, as specified in section 2.1.1. It is important to consider that OpenPGP certificates can contain multiple subkeys, and this may easily generate certificates in the tens of megabytes.

Finally, BIKE [ABB+22a] is a KEM based on binary linear Quasi-Cyclic Moderate Density Parity Check (QC-MDPC) codes. Its performance is comparable to the already selected CRYSTALS-Kyber, and its artifact size is just slightly larger, specifically in the range of 3 to 5 KB for public keys and ciphertexts of security level 3 and 5. If a standardization body, such as NIST or the CFRG, were to standardize this algorithm,

TU WIEN Bibliothek Your knowledge hub
Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

it could be a great candidate for an alternative KEM based on a different underlying problem. Given the great legacy burden of OpenPGP we do not consider adding non-standardized algorithms, since their implementations becomes un-modifiable and support can not be easily dropped. An unfortunate example can be found with the deployment of Curve25519, used in OpenPGP before the release of the X25519 standard RFC 7748 [LHT16]: OpenPGP is now stuck with two different variants of X25519, one of which presents a unique and non-standard way to encode artifacts.

2.3 Key Derivation and Combination

To formally construct a secure hybrid KEM as proposed in this project, we need to define all its building blocks. This section will provide a definition of KEM, introduce the concept of Indistinguishability under adaptive Chosen Ciphertext Attack (IND–CCA2) security, and how to build a Key Derivation Function (KDF) to meet this definition. Finally we will determine how to combine two or more KEMs such that if an adversary has full control over all but one of the secret shares, the combination is still secure.

The construction laid out in this section and used in the project is compliant with NIST SP800-56C [BCD20], is proven secure in the paper "KEM Combiners" by Giacon, Heuer, and Poettering [GHP18], and directly based on the Internet Research Task Force (IRTF) draft standard `draft-ounsworth-cfrg-kem-combiners-03` [OWK23], of which I am also an author. Since only Secure Hash Algorithm 3 (SHA–3) and Keccak Message Authentication Code (KMAC) hash functions are used in the proposed construction, we will focus on these and do not analyze hash construction in general.

2.3.1 Key Encapsulation Mechanisms

A KEM [KL20] is a tuple of probabilistic polynomial-time algorithms (Gen, Encaps, Decaps) such that:

1. The key-generation algorithm Gen takes as input the security parameter 1^n and outputs a public-/private-key pair (pk, sk). We assume pk and sk each has length at least n, and that n can be determined from pk. We write this as
$$(pk, sk) \leftarrow \mathrm{Gen}(1^n).$$
2. The encapsulation algorithm Encaps takes as input a public key pk (which implicitly defines n). It outputs a ciphertext c and a key $k \in 0, 1^{\ell}(n)$ where ℓ is the key length. We write this as
$$(c, k) \leftarrow \mathrm{Encaps}_{pk}(1^n).$$
3. The deterministic decapsulation algorithm Decaps takes as input a private key sk and a ciphertext c, and outputs a key k or a special symbol $\perp$ denoting failure. We write this as
$$k := \mathrm{Decaps}_{sk}(c).$$

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

It is required that with all but negligible probability over the randomness of Gen and Encaps, if $\text{Encaps}_{pk}(1^n)$ outputs (c, k) then $\text{Decaps}_{sk}(c)$ outputs k.

The typical use of KEMs is to establish a shared secret between the sending party, or *encapsulater*, and the receiving party, or *decapsulater*.

2.3.2 Chosen-Ciphertext Attack Security

In order to construct a secure KEM we need to define the notion of IND–CCA2 security. A scheme is secure in this definition if no adversary can distinguish the ciphertext of two messages, despite having access to a decryption oracle that they are not allowed to use on the challenge ciphertext.

Formally, let $\mathcal{A}$ be an adversary and let $\Pi = (\text{Gen}, \text{Encaps}, \text{Decaps})$ be a KEM with key length n, and consider the following experiment [KL20].

The adaptive Chosen Ciphertext Attack (CCA) indistinguishability experiment $KEM^{cca}_{\mathcal{A},\Pi(n)}$:

1. $\text{Gen}(1^n)$ is run to obtain keys (pk, sk). Then $\text{Encaps}_{pk}(1^n)$ is run to generate (c, k) with $k \in \{0, 1\}^n$.
2. Choose a uniform bit $b \in \{0, 1\}$. If $b = 0$ set $\hat{k} := k$. If $b = 1$ then choose a uniform $\hat{k} \in \{0, 1\}^n$.
3. $\mathcal{A}$ is given (pk, c, $\hat{k}$) and access to an oracle $\text{Decaps}_{sk}(\cdot)$, but may not request decapsulation of c itself.
4. $\mathcal{A}$ outputs a bit b'. The output of the experiment is defined to be 1 if $b' = b$, and 0 otherwise.

A KEM Π is CCA secure if for all probabilistic polynomial-time adversaries $\mathcal{A}$ there is a negligible function negl such that:

$$\Pr[KEM^{cca}_{\mathcal{A},\Pi(n)} = 1] \leq \frac{1}{2} + \text{negl}(n)$$

Using a CCA secure KEM in combination with a CCA secure private-key encryption scheme results in a CCA secure public-key encryption scheme. Given that the allowed combination of packets permits only usage of integrity protected data, also CCA secure, we wish to preserve this notion when deriving and combining the KEK.

2.3.3 Keccak

Keccak is based on random sponge functions [BDPvA11], which provide arbitrary output length, and behave like a random oracle, except for the effects induced by the finite memory.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

The sponge construction is a simple iterated construction for building a function F with variable-length input and arbitrary output length based on a fixed-length transformation or permutation f operating on a fixed number b of bits. Here b is called the width. The sponge construction operates on a state of $b = r + c$ bits. The value r is called the bitrate and the value c the capacity.

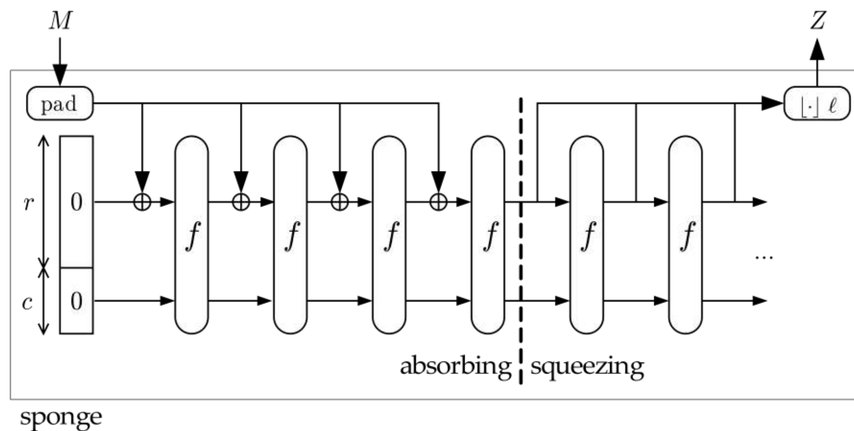


Figure 2.8: Random sponge function construction.

The following procedure is illustrated in fig. 2.8. First, all the bits of the state are initialized to zero. The input message is padded and cut into blocks of r bits. The sponge construction then proceeds in two phases: the absorbing phase followed by the squeezing phase.

- In the absorbing phase, the r-bit input message blocks are XORed into the first r bits of the state, interleaved with applications of the function f. When all message blocks are processed, the sponge construction switches to the squeezing phase.
- In the squeezing phase, the first r bits of the state are returned as output blocks, interleaved with applications of the function f. The number of output blocks is chosen at will by the user.

The last c bits of the state are never directly affected by the input blocks and are never output during the squeezing phase.

In Keccak's instantiation, the function f consists of $12 + 2\ell$ rounds, each composed of five steps:

1. θ: A linear mapping that aims at maximising diffusion. Without it, f would not provide diffusion of any significance.
2. ρ: A linear mapping that consists of translations within the lanes aimed at providing inter-slice dispersion. Without it, diffusion between the slices would be very slow.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

3. π: A transposition of the lanes that provides dispersion. Without it, f would exhibit periodic trails of low weight.

4. χ: Is a non-linear transformation composed of S-boxes. Without it, f would be linear.

5. ι: The addition of round constants, aimed at disrupting symmetry. Without it, f would be translation-invariant in the z-direction and all rounds would be equal.

We rely on Keccak's properties in several components of the construction. In fact, for given capacity c the indifferentiability proof shows that assuming there are no weaknesses found in the Keccak permutation, an attacker has to make an expected number of $2^{c/2}$ calls to the permutation to tell KMAC from a random oracle. For a random oracle, a difference in only a single bit gives an unrelated, uniformly random output [BDPvA11].

2.3.4 Keccak Message Authentication Codes

A KMAC is a variable-length Message Authentication Code (MAC) algorithm based on Keccak that can also be used as a PRF. It is defined in SP800-185 [KjCP16], and differs from the other Keccak-based PRFs SHAKE and cSHAKE because altering the requested output length generates a new, unrelated output.

KMAC takes the following parameters:

- The key `K`, a bit string required to be at least as long as the security strength for its usage to be approved.
- The main input bit string `X`.
- The output bit length `L`.
- The optional customization string `S`.

KMAC exists in two flavors, KMAC128 and KMAC256, instantiating Keccak with capacity c respectively 256 and 512, and rate b respectively 168 and 136. In this project only KMAC256 is used, and it is constructed as follows:

```
encode_string(S) = left_encode(len(S)) || S
X' = bytepad(encode_string(K), 136) || X || right_encode(L)
C = bytepad(encode_string("KMAC") || encode_string(S), 136)
return KECCAK[512](C || X' || 00, L)
```

Where the `left_encode` and `right_encode` functions encode integers up to $2^{2040} - 1$ from the left or from the right as strings.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

2.3.5 Key Derivation Functions

In the scope of the project an IND–CCA2 secure KEM is built out of the ECDH algorithm. To achieve this, we use the Hashed ElGamal key encapsulation method: the encryption key will be derived by hashing a pair of group elements of the elliptic curve field:

$$\mathrm{KDF}_{dk}^{\lambda,\Gamma}(a,b) := \mathrm{Hash}(a||b)$$

Where:

- a is the auxiliary group element, in our case the encoded Elliptic Curve (EC) public point in the ephemeral exchange.
- b is the shared secret, in our case the x-coordinate of an EC point.
- dk is a derivation key, an element in the KDF's key space.
- $\lambda \in \mathbb{Z}_{\geq 0}$ is the security parameter.
- $\Gamma \in [S_\lambda]$ is a group description that specifies a finite abelian group $\hat{G}$, along with a prime order subgroup G, a generator g of G and the order q of G, all public parameters of the curve.
- $||$ represents the concatenation operator.

A KDF is secure if the distinguishing advantage between the distribution of $\mathrm{KDF}_{dk}^{\lambda,\Gamma}(a,b)$ and the distribution of a random key K is negligible. This has been proven to be the case when the KDF is pairwise independent. This is defined formally as $\forall a, b, b' \in G$ with $b \neq b$, the distribution

$$\left\{\left(\mathrm{KDF}_{dk}^{\lambda,\Gamma}(a,b), \mathrm{KDF}_{dk}^{\lambda,\Gamma}(a,b')\right) : dk \leftarrow \mathrm{KDF}.\mathrm{KeySpace}_{\lambda,\Gamma}\right\}$$

is the uniform distribution over all pairs of bits strings of length $\mathrm{KDF}.\mathrm{OutLen}(\lambda)$ [CS03].

2.3.6 Key Combiners

In the scope of hybrid PQ cryptography, another issue consists in combining two or more secrets preserving the security guarantees, in our case IND–CCA2 security.

In order to achieve this, we will construct a key combiner derived from the paper "KEM Combiners" by Giacon, Heuer, and Poettering [GHP18], in particular the combiner in example 3 of figure 1, based on a Split-Key Pseudo-Random Function (SKPRF). We start by defining an SKPRF, then assuming there are no weaknesses found in the Keccak permutation prove that KMAC is one, and finally show that our construction is equivalent to example 3.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Practically, an SKPRF is the extension of a dual-PRF that can be keyed by any of its inputs. This means, given an adversary that controls all the keys but one that is picked uniformly at random, the function still behaves like a random function.

Formally, given finite key spaces $\mathcal{K}_1, \dots, \mathcal{K}_n$, an input space $\mathcal{X}$, and a finite output space $\mathcal{Y}$, let $\mathcal{K} = \mathcal{K}_1 \times \cdots \times \mathcal{K}_n$, and consider a function $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$. We introduce the following game $\mathrm{PR}_i^b(\mathcal{A})$ with $b \in \{0,1\}$ and $1 \le i \le n$:

1. Let X be the empty set.
2. Randomly sample $k_i \xleftarrow{\$} \mathcal{K}_i$.
3. Query the adversary $\mathcal{A}$ for b'
4. The output of the experiment is defined to be 1 if $b' = b$, and 0 otherwise.

Using the following oracle $\mathrm{Eval}(k', x)$:

1. The oracle aborts if $x \in X$.
2. $X \leftarrow X \cup \{x\}$.
3. The values of all $k_j\ \forall j \neq i$ are set by the adversary, $k_1, \dots, k_{i-1}, k_{i+1}, \dots, k_n \leftarrow k'$.
4. The value y^0 is obtained by evaluating F, that is $y^0 \leftarrow F(k_1, \dots k_n, x)$.
5. The value y^1 is randomly sampled from $\mathcal{Y}$, formally $y^1 \xleftarrow{\$} \mathcal{Y}$.
6. y^b is returned.

For each index $i \in [1 \cdots n]$, we associate with an adversary $\mathcal{A}$ its advantage:

$$\mathrm{Adv}_{F,i}^{\mathrm{pr}}(\mathcal{A}) := |\Pr[\mathrm{PR}_i^0(\mathcal{A})] = 1] - \Pr[\mathrm{PR}_i^1(\mathcal{A})] = 1]|.$$

Observe that, for any index i, in the game $\mathrm{PR}_i^b, b \in \{0,1\}$, the i-th key component of F is assigned at random in point (1) of the game definition, while the adversary contributes the remaining $n-1$ components on a per-query basis, as per point (3) of the oracle definition. We say that F is an SKPRF if the advantages $\mathrm{Adv}_{F,i}^{\mathrm{pr}}$ for all key indices are negligible for all practical adversaries.

Given a KMAC instance as defined in section 2.3.4 we now evaluate it on the input string $s_1 || \dots || s_n$, where $||$ represents concatenation. Assuming there are no weaknesses found in the Keccak permutation, for each bit changed in the concatenated input we obtain a uniformly random output, as shown in section 2.3.3. Given s obtained from the application of Keccak on the concatenation of strings $s_1, \cdots s_n$, an adversary has to correctly guess all strings entirely in order to be able to distinguish s from an uniformly

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

random string. Following the previous definition, adversary has therefore negligible advantage in guessing the correct b'.

Among the one-step KDF standardized by NIST SP800-56C [BCD20] we will evaluate the following:

```
KMAC256(salt, x, outputBits, "KDF")
```

Where

- `salt` represents a non-null agreed upon string.
- `x` is the input formatted as `counter || Z || FixedInfo`. Here, `Z` is the concatenation of the ciphertexts and shared secrets `c_1 || ss_1 || ... || c_n || ss_n`.
- `outputBits` is the required number of key material bits.

Considered the construction proposed in example 3 [GHP18]

$$H(k_1, \ldots, k_n, c),$$

where k_1 to k_n are the shared secrets, and c is the concatenated ciphertext, we can argue that this is equivalent to the KDF proposed in the previous paragraph, in fact:

- Under the given assumptions KMAC behaves like a SKPRF.
- A `counter` prefix and a `FixedInfo` suffix, constant for each instance, do not compromise the SKPRF property.
- The two constructions are simply a reordering of the arguments, therefore equivalent.

Therefore we have that

$$\begin{aligned} H(k_1, \ldots, k_n, c) &= \text{KMAC256}(k_1 || \ldots || k_n || c_1 || \ldots || c_n) \\ &\iff \text{KMAC256}(c_1 || k_1 || \ldots || c_n || k_n) \\ &\iff \text{KMAC256}(\text{counter} || c_1 || k_1 || \ldots || c_n || k_n || \text{FixedInfo}) \end{aligned}$$

where only the main input bit string `X` is here considered in the arguments of the function KMAC256(·).

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

CHAPTER 3

Protocol Design Changes

When embedding PQ cryptography into an existing protocol, several design decisions have to be taken, that are a trade off between security, implementation complexity, crypto-agility, and usability.

As mentioned in the introduction, this project was born joining two different projects at the IETF 113, where a presentation about the BSI-sponsored project was held. Both projects had the same objective of implementing PQ cryptography into OpenPGP, but with completely different design principles.

In this chapter the various design choices are going to be discussed and justified. We will start from an overview of the high-level design, such as composite vs composable, then present a concrete set of cryptographic algorithms chosen based on the theoretical background illustrated in chapter 2, and finally explain the small bits and pieces.

All the high-level protocol design changes have been extensively discussed with the OpenPGP community at the OpenPGP Email Summit 2022[1], at the IETF 115 in London [EHK+22], and on the mailing list[2]. This was done with the intention of gathering feedback directly from the implementers, to ensure a smooth transition to the standardization phase.

3.1 Composite vs Composable

While there seemed to be consensus for implementing the PQ algorithms in hybrid mode with elliptic curves, as recommended from the BSI [EHH+20] as well as the authors of CRYSTALS-Kyber and CRYSTALS-Dilithium, the two initial projects did not agree on how flexible should this combination mechanism be.

[1] Minutes can be found at https://wiki.gnupg.org/OpenPGPEmailSummit202205Notes

[2] Historical archives can be found at https://mailarchive.ietf.org/arch/browse/openpgp/

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Composite schemes are designed to have a fixed combination of two or more schemes, coming as an an indivisible package. From the implementation perspective, the composite model is simpler, because it appears as a single algorithm on the protocol layer, i.e. the current OpenPGP algorithm structure can be reused. The implementation is agnostic to the cryptographic primitive, as long as it can offer a KEM interface to encapsulate and decapsulate a key, or a signer interface to sign and verify. Furthermore, composite schemes have a simpler cryptographic construction: for the considered algorithms all the artifacts have a fixed length, ensuring a straightforward data encoding and a simpler KDF construction.

On the other hand Composite schemes allow very little crypto-agility. Creating a new combination requires standardization of a new algorithm, update of the deployed software, and generation of new keys, procedure that might last years in the OpenPGP context.

Composable schemes instead are designed to allow a user to mix a variable combination of algorithms to create a hybrid scheme. The implementation may choose any of the available algorithms as building block and concatenate the data, to allow the creation of a new composable packet. This approach maximizes crypto-agility, allowing the users to create keys with a very high degree of flexibility, as any set of existing or new algorithms can be used to create a key.

With great power comes great responsibility, and this approach carries some complexity drawbacks:

- **Algorithm security policy**: Public key users have to decide on a policy of what is an acceptably secure algorithm. For instance, any modern OpenPGP implementation would reject verifying a signature created with an RSA-512 key. With composable schemes, a similar policy needs to be created and kept consistent across implementations. Using fixed combination ensures a clear communication, e.g. asserting that Ed25519 as standalone is no longer accepted, while Ed25519 + Dilithium3 is. Using variable combinations requires a more complex policy, where individual algorithms are not be accepted, but specific combinations are. It is important to note that this scheme list may grow factorially with the number of supported algorithms.

- **Implementation complexity**: Following the composite approach adds another layer between the protocol layer and the algorithms layer. In fig. 3.1 a concrete example of the changes required to implement a composite or composable PKESK is illustrated. In fig. 3.1a we can observe that no changes on the protocol layer are required to implement a composite scheme, since it appears no different than the existing X25519 packet. In the algorithm-specific part there are two ciphertexts instead of one. In fig. 3.1b are instead shown the changes required for a composable scheme, where a generic PKE algorithm ID is introduced, creating a further wrapping layer around the ciphertexts.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

- **Interoperability**: Fixed algorithms provide a simple interoperability testing framework. The OpenPGP community is actively working on an interface to allow simple interoperability testing called Stateless OpenPGP Protocol (SOP) [Gil22], that is used to test the handling of common and edge cases across different implementations[3]. Having mixed combinations would require a more complex and computationally expensive interoperability testing suite, in an environment where interoperability is already precarious.

- **Downgrade attacks**: keys used in a composable context must be distinguishable from keys used in a traditional context, i.e. an adversary should not be able to perform a downgrade attack by reusing part of a PQ key or ciphertext. For example, an attacker may try to remove the PQ component from a composable key: the parsing implementation must refuse that key as invalid. This criteria adds complexity when creating backward-compatible PQ keys, that might be used to encrypt communication both traditionally or in hybrid mode depending on the capabilities of the implementation sending the message.

  For instance, one may try to design a change to the OpenPGP protocol that issues two separate keys, ECDH and PQ, where a legacy implementation may use only the ECDH key, while a newer implementation must use the key material from both to encapsulate a session key. In this case, a signalling mechanism ignored by the legacy implementation but enforced from the newer implementations is required. This procedure, done via non-critical subpackets on the key's self-signature, is illustrated in fig. 3.2b. This adds a cross-layer dependency that significantly complicates the implementation.

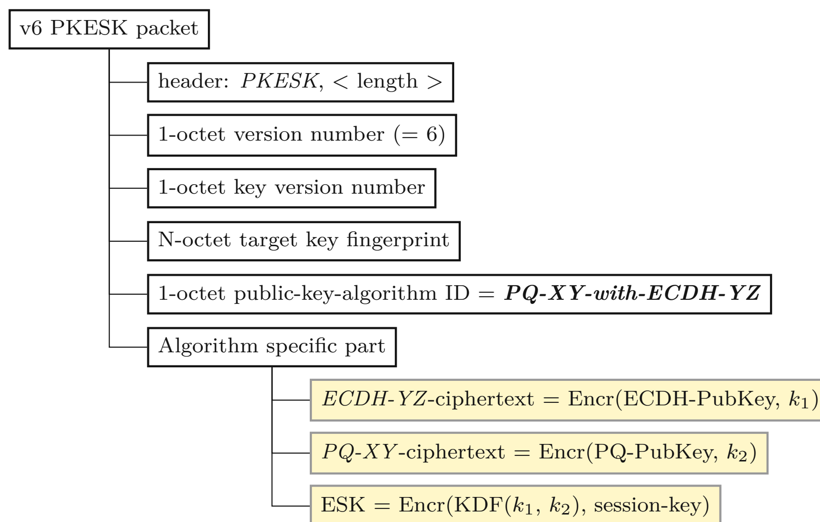
  In the composite case, see fig. 3.2a, there is no backward-compatibility option. It is only possible to generate two independent public keys, one for legacy implementations and one for PQ implementations. More details on the planned migration in this case follow in section 3.6.

Finally, if an implementation limits the flexibility by reducing the amount of possible combinations using a whitelist to address the interoperability or security policy concerns, the composable approach loses any practical advantage compared to the composite schemes, as allowing a new combination requires update and deploy of all implementations, but retains the implementation complexity disadvantages.

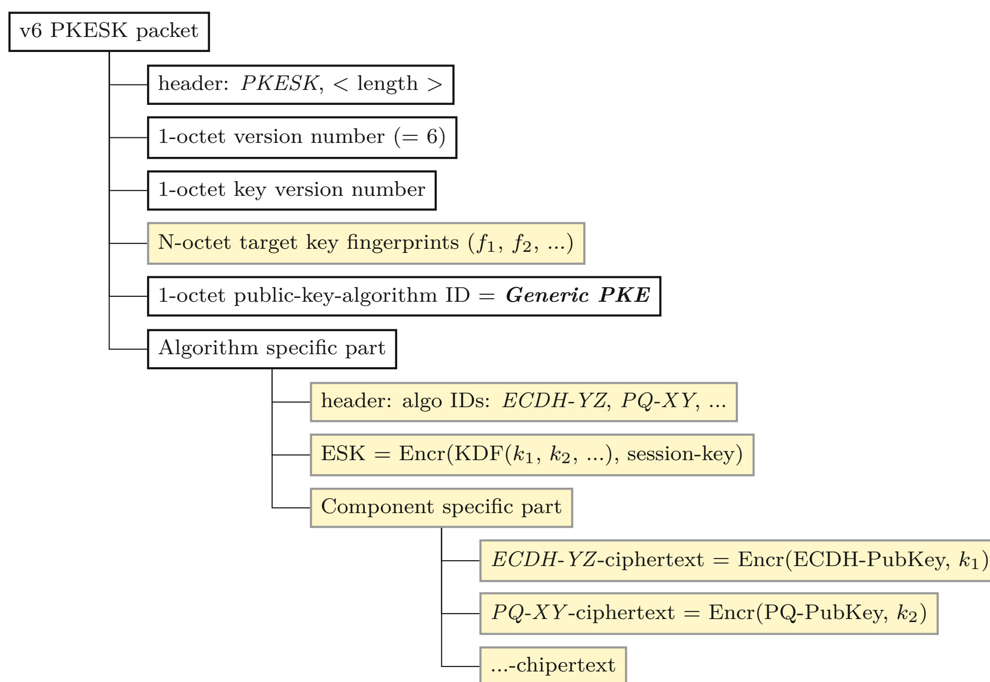
For all these reasons, it was decided to follow the composite approach, and attempt standardization with fixed algorithm combinations. This approach was also discussed with the community at the May 2022 OpenPGP summit, that generally agreed.

[3] More info and results can be found at `https://tests.sequoia-pgp.org/`

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub



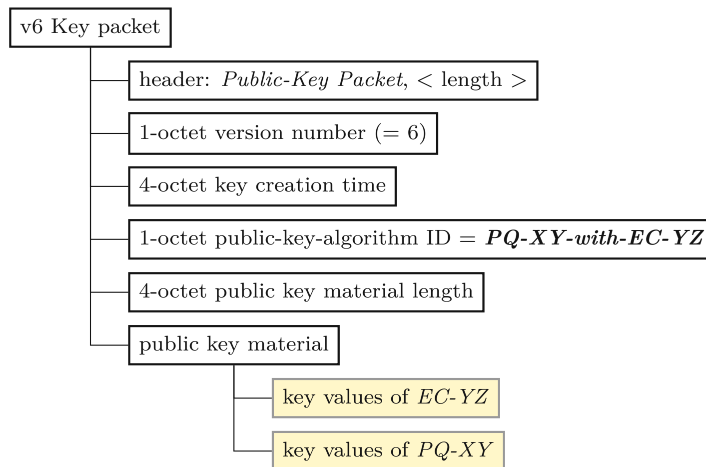
(a) Composite PKESK packet structure: no changes on the protocol layer, a composite scheme appears as a single algorithm ID for OpenPGP.



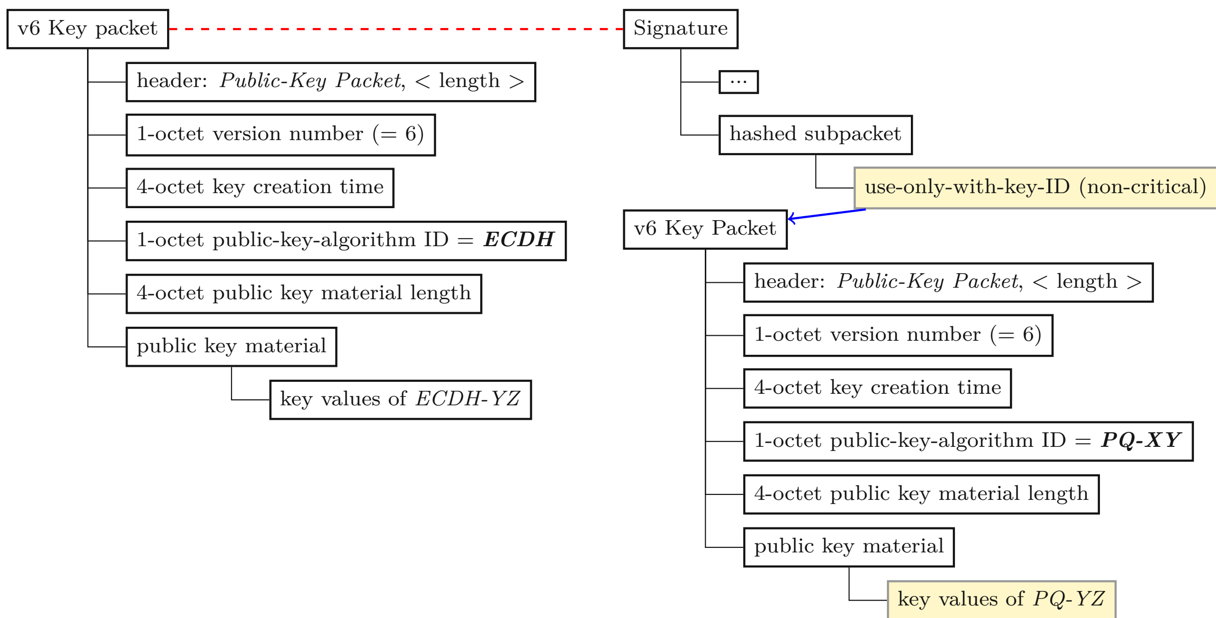
(b) Composable PKESK packet structure: creation of a generic algorithm identifier that has an arbitrary algorithm composition.

Figure 3.1: Composite vs Composable PKESK packet structure. Changes highlighted in yellow.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub



(a) Composite key packet structure: no changes on the protocol layer, a composite scheme appears as a single algorithm ID for OpenPGP.



(b) Possible backwards composable key structure, where the PQ key material is linked to an existing key via a reference in the self-signature or subkey binding signature. Note that this model breaks the current OpenPGP layering by requiring key material from multiple keys to decrypt a message or verify a signature.

Figure 3.2: Composite vs Composable key packet structure. Changes highlighted in yellow.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

3.2 Key Encapsulation Mechanisms

Recalling from section 2.1.2, OpenPGP public key encapsulation works in the following way:

1. A session key is randomly generated or taken as input.
2. A PKESK is created for each recipient, asymmetrically wrapping the session key.
3. The PKESKs are prepended to the encrypted data.

The session key can also be taken as input because it is not always desirable to have a fresh one: sometimes it is computationally expensive to re-encrypt the body of the message, as it might be several gigabytes large, or there is no intention on changing the SEIPD; session key reuse is a quite common case in OpenPGP. This constraint would introduce an additional abstraction layer in the composable model, but plays well with the composite structure, ensuring a simple yet robust construction, where composite schemes appear as a single algorithm on the protocol layer. The proposed structure of a PKESK is illustrated in fig. 3.1a.

The complete list of algorithm identifiers to algorithm association is presented in table 3.1. This has been discussed with several stakeholders in the protocol that had to conform to different regulatory requirements, and needed optional algorithm IDs for compliance. The requirement levels differ because we need to ensure a minimal interoperability option, Kyber in combination with X25519, recommend a higher security option, Kyber in combination with X448, and provide an optional set of algorithms for compliance-specific cases.

ID	Algorithm	Requirement
29	Kyber768 + X25519	MUST
30	Kyber1024 + X448	SHOULD
31	Kyber768 + ECDH–NIST–P–256	MAY
32	Kyber1024 + ECDH–NIST–P–384	MAY
33	Kyber768 + ECDH–brainpoolP256r1	MAY
34	Kyber1024 + ECDH–brainpoolP384r1	MAY

Table 3.1: Proposed KEM algorithm IDs and their implementation requirement.

3.2.1 ECC Key Share Derivation

While Kyber's proposed construction is already IND–CCA2 secure, the elliptic curve counterpart is not. In particular, since there can be multiple points on the curve that

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

correspond to the same secret scalar, an adversary that can query a decryption oracle for P' such that $P' \equiv P \equiv xG$, with $P' \neq P$, can obtain the shared secret for the point P [LHT16], therefore breaking CCA security, as discussed in section 2.3.2.

To prevent this attack, it was decided to rely on the hashed ElGamal key encapsulation construction as discussed in section 2.3.5, where the ciphertext is hashed with the shared secret obtained by the exchange:

$$K_E = \text{hash}(SS_E || C_E)$$

Where SS_E is the x-coordinate derived from the exchange, and C_E is the encoded ephemeral public point of the exchange. The hash function family is SHA–3, and the size depends on the algorithm ID.

Given that our construction relies only on the Keccak-based SHA–3 hash function, we can then show that this makes the KDF pairwise independent. In fact, assuming there are no weaknesses found in the Keccak permutation, for each bit changed in the concatenated input we obtain a uniformly random output, as shown in section 2.3.3. This satisfies the definition of pairwise independence given in section 2.3.5, therefore our construction is IND–CCA2 secure.

In practice, embedding the ciphertext into the hash prevents malleability, ensuring that that any adversarial change to the ciphertext diffuses uniformly on the key share.

3.2.2 Key Derivation Function

A KDF derives a KEK from the traditional and PQ key shares. In fig. 3.1a the KDF(k_1, k_2) function is mentioned: this key component of the algorithm is critical for a secure construction, since it provides key share combination, public key binding, protection against CCA attacks, and domain separation.

The following KMAC-based design is used:

```
//   multiKeyCombine(eccKeyShare, eccCiphertext, kyberKeyShare,
                     kyberCiphertext, fixedInfo)

//   Input:
//   domSeparation - the UTF-8 encoding of the string
//                   "OpenPGPCompositeKeyDerivationFunction"
//   counter - a 4 byte counter set to the value 1, that is
//             the value 0x00000001
//   K_E - the ECC key share encoded as an octet string
//   C_E - the ECC ciphertext
//   K_K - the Kyber key share encoded as an octet string
//   C_K - the Kyber ciphertext
//   algID - the algorithm ID encoded as octet
```


Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

```
//   publicKey - the recipient's encryption sub-key packet
//                  serialized as octet string
//   oBits - the size of the output keying material in bits
//   customization - the UTF-8 encoding of the string "KDF"

// Output:
// K - A Key Encryption Key to wrap the session key

fixedInfo = algID || SHA3-256(publicKey)
encKeyShares = counter || K_E || C_E || K_K || C_K || fixedInfo
K = KMAC256(domSeparation, encKeyShares, oBits, customization)
```

This design, inline with NIST recommendation SP800-56C [BCD20], was chosen after considering several designs using Hash-based Message Authentication Code (HMAC) [ADK+22] or SHA–3 concatenation, because it ensures the following properties:

- **Secure key share combination**: if an adversary has full control of one key share, no information is revealed about the other. Since KMAC is based on Keccak, we can then state that assuming there are no weaknesses found in the Keccak permutation, to be able to distinguish a key K, derived from the application of KMAC to the concatenation the shared keys K_E and K_K from a random bit string, an adversary has to correctly guess both key shares K_E and K_K entirely, as shown in section 2.3.6.

- **Public key binding**: safely bind the KEK to the public keys used in the original encryption process, preventing ciphertext manipulation, such as proxy transformations. In order to obtain the same KEK, the recipient needs to provide the expected encryption `publicKey` and the encoded ciphertexts that are currently being used for decryption. The serialization of these is univocally standardized by the OpenPGP protocol. Any attempt at performing a cross-algorithm attack, or proxy transformations that the recipient is not aware of, will result in an invalid ciphertext. The recipient's public key is embedded in the `fixedInfo` parameter, and to keep its length constant it is hashed, fixing the position of the algorithm ID from the end. The sender's public key is in this context represented by the two ephemeral ciphertexts included in the KDF, since the sender may not have an OpenPGP key.

- **Preserving IND–CCA2 security**: in the proposed construction we combine the ephemeral ciphertexts into the KDF, ensuring that if at least one of the ingredient KEM is IND–CCA2 secure, then the combination is secure, as stated in section 2.3.6. Effectively, this means we are including the ciphertexts in the KDF a second time to ensure a solid security proof of the construction, even if both key shares already include them in the derivation.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

- **Domain separation**: Ensures that no other algorithm builds the KEK in the same way, preventing cross-algorithm attacks. A source of cross-algorithm attacks can also be created from the existing ECDH KDF, that has a variable length `fixedInfo` format, but compatible parameters, as it may use SHA–3 for the KEK derivation. For this reason, we opted for a KMAC-based construction instead of plain SHA–3, using a fixed salt as key.

The derived KEK is then used to encrypt the provided session key using 256-bit Advanced Encryption Standard (AES) as standardized in RFC 3394 [HS02].

3.2.3 Multiple PKESK

When sending a message to multiple recipients, the same session key is wrapped for each one of them. It is important to note, that as long as there is one recipient not supporting PQ algorithms, then the message is still effectively not PQ protected. The proposed specification therefore forbids applications from encrypting a message for the same recipient in both a PQ and traditional way.

3.3 Signature Schemes

We introduce in this project three different signature schemes: Dilithium with EdDSA, Dilithium with ECDSA, and SPHINCS$^+$. As recommended from the authors, the first two are in combination with EC signatures, to ensure a fallback in case of significant new attacks on Dilithium.

The complete list of algorithm identifiers to algorithm association is presented in table 3.2. Similarly to the KEM schemes, it has been negotiated with all the stakeholders to conform to different regulatory requirements. Also here, we have Dilithium3 in combination with Ed25519 mandatory, Dilithium5 in combination with Ed448 and SPHINCS$^+$–SHA2 recommended, while all the compliance-specific algorithms are optional.

Similarly to Kyber, also Dilithium signature schemes have been built with a composite construction, and therefore require no change on the protocol level. OpenPGP considers the composite signature scheme as a single algorithm, and we require both component signatures to successfully verify in order to consider the signature valid. An example of composite signature structure is illustrated in fig. 3.3.

SPHINCS$^+$ instead is considered mature enough for standalone standardization. Its security analysis [ABB$^+$22b] includes a complete history of third party cryptoanalysis with estimated security strength and known attacks. The OpenPGP community agreed with the previous statement, and we therefore propose to not combine the algorithm with elliptic curves. To prevent an explosion of the number of possible algorithms it was decided to parametrize SPHINCS$^+$ for speed and security trade-offs. Parameter values are listed in table 3.3.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

ID	Algorithm	Requirement
35	Dilithium3 + Ed25519	MUST
36	Dilithium5 + Ed448	SHOULD
37	Dilithium3 + ECDSA-NIST-P-256	MAY
38	Dilithium5 + ECDSA-NIST-P-384	MAY
39	Dilithium3 + ECDSA-brainpoolP256r1	MAY
40	Dilithium5 + ECDSA-brainpoolP384r1	MAY
41	SPHINCS+–simple–SHA2	SHOULD
42	SPHINCS+–simple–SHAKE	MAY

Table 3.2: Proposed signature algorithm IDs and their implementation requirement.

Parameter ID	Parameter
1	SPHINCS+–simple–*–128s
2	SPHINCS+–simple–*–128f
3	SPHINCS+–simple–*–192s
4	SPHINCS+–simple–*–192f
5	SPHINCS+–simple–*–256s
6	SPHINCS+–simple–*–256f

Table 3.3: Proposed SPHINCS$^+$ parametrization.

3.3.1 Multiple Signatures

Signature schemes differ from PKE because there is an advantage in having two signatures, PQ and traditional, issued from the same user. As seen in section 2.1.2, these can in fact be concatenated on in a message to provide forward security. The OpenPGP standard allows having several signatures, considering a message to be signed when at least one verifies. It is up to the receiving implementation whether to accept a traditional signature: this could allow a deadline-based transition or just deprecation of traditional signatures in future versions. Given this paradigm, the proposed specification recommends to sign a message multiple times, once with a PQ signature, and once with a traditional EC signature, shifting the acceptance policy to the receiver. This ensures maximal compatibility without any authenticity trade-off.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

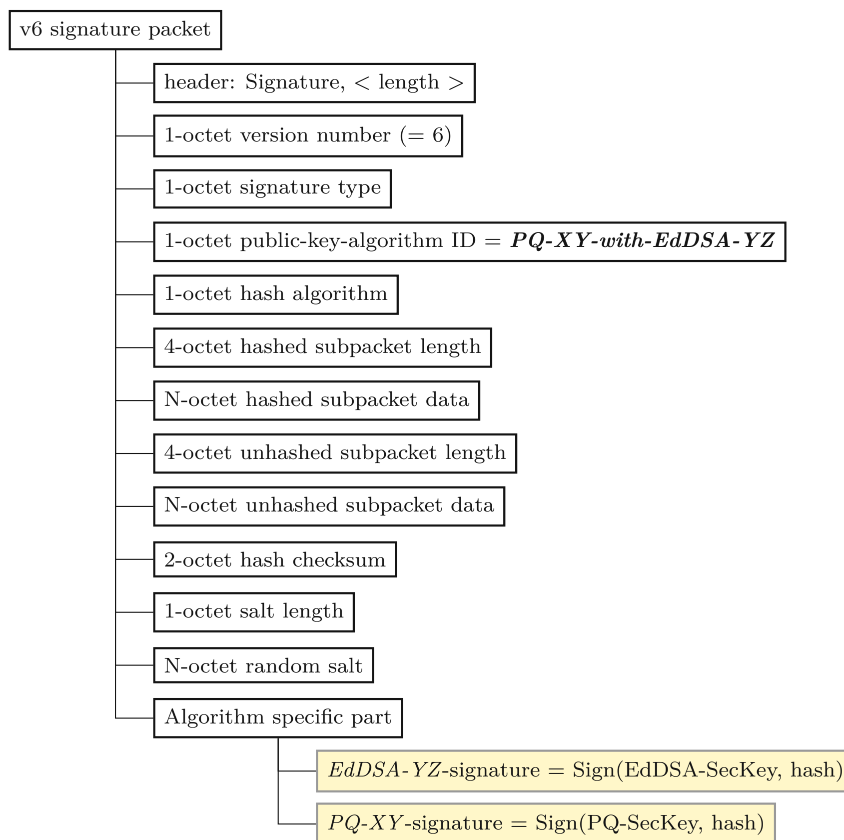


Figure 3.3: Composite signature packet. Changes highlighted in yellow.

3.3.2 Hash Function Binding

Since OpenPGP pre-hashes the data on the protocol layer, the PQ draft specification proposes to bind the hash function used in the digest generation to the hash function used internally in the algorithm, e.g. since Dilithium produces internal digests using SHAKE, the attack surface would be expanded by using SHA–2 in the message digest generation.

For PQ algorithms, we propose the bindings listed in table 3.4. Dilithium uses internally a SHAKE PRF, therefore we bind it to either SHA–3 256 or 512 bit, depending on the implementation's security settings. SPHINCS$^+$, instead uses different hashes depending on the parameters, therefore a more sophisticated binding is necessary.

It was considered to alternatively remove the hash-then-sign paradigm altogether and feed the PQ algorithms directly the message as input, but this would have required extensive changes to the OpenPGP protocol. It is important to note, that OpenPGP allows streaming, and therefore the implementation might not have access to the data multiple times to rewind and perform a second hashing. The required data for the instantiation are embedded in the signature packet, that always follows the signed data. Considered

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Algorithm ID	Parameter ID	Hash function	Hash function ID
31 - 36	N/A	SHA3–256, SHA3–512	12, 14
37	1, 2	SHA–256	8
37	3, 4, 5, 6	SHA–512	10
38	1, 2	SHA3–256	12
38	3, 4, 5, 6	SHA3–512	14

Table 3.4: Proposed signature hash binding.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

that PQ algorithms are bound to use version 6 signatures, featuring salted hashing as stated in section 3.5, the PQ draft specification proposes to keep the hash-then-sign paradigm. This is not a security tradeoff, since in SPHINCS$^+$ the salt length matches the salt size in the internal construction. It is to be noted that binding the salt size to the hash function is not specific to PQ algorithms and involves a protocol change, therefore instead of limiting the scope to the PQ draft specification, this binding was integrated directly into the current standard development process [WHWY23].

3.4 Artifact Encoding

This project uses a new encoding for key material, ciphertext and signature artifacts in OpenPGP: fixed-length octet strings. Even though this seems to be the easiest data format, it is historically not used to store these artifacts: OpenPGP was born with RSA and DSA in mind, and therefore it is using Multi-Precision Integer (MPI) encoding. While this is very suitable to store integers, it has been reused to store encoded EC points for algorithms such as ECDH, ECDSA, and EdDSA. In order to avoid truncation when encoding as MPI, points on the curve are prefixed with a `0x40` or `0x80` leading byte.

Since the NIST competition [MAA+22] requires all candidates to offer only an opaque fixed-length octet string for all artifacts, it was decided to ditch the MPI encoding in favor of less flexible and simpler system.

Given that the composite model was chosen, cryptographic material is encoded only in the algorithm-specific part, like shown in fig. 3.2a.

3.5 Version Binding

The PQ draft specification requires version 6 key, PKESK, signature, and OPS packets when using PQ algorithms. This improves security providing the following advantages:

- SHA–1 is not used to produce signatures or fingerprints. The latest OpenPGP specification explicitly forbids using SHA–1 when generating the message digest for

TU WIEN Bibliothek Your knowledge hub

signature, and for v6 keys replaces it with SHA–256 in fingerprint generation.

- Prevents ciphertext malleability attacks. A version 6 PKESK is always followed by a version 2 SEIPD that features only AEAD protected symmetric cryptography modes.
- SHA–3 support. Optional support for SHA–3 was introduced in the latest OpenPGP specification only. This project proposes to make it mandatory, and binding the use of SHA–3 to the relevant PQ algorithms.
- Salted signatures. OpenPGP uses a hash-then-sign paradigm on the protocol level, in order to implement easier data streaming. In particular, an OPS precedes the data, informing that a signature follows, and the signature details are at the end of the data stream. Version 6 signatures include an unpredictable salt in the OPS packet, that is prepended to the hashed data, ensuring that weak-collision resistance is sufficient for the hash function used in the data hashing [LP20] as analyzed in section 2.1.2.

3.6 Proposed Migration Strategies

We propose two different migration strategies, depending on the use case.

1. The first strategy is based on having two keys: PQ and traditional. Older clients will be able to use the traditional key, while newer clients can prefer the PQ key. To enhance this approach it could be interesting to enhance the specification with a a “key superseded” signature, that allows to designate a replacement key without effectively revoking the old one.
2. The second strategy consists of attaching a PQ encryption subkey to a classical existing key. By design, we require implementations to prefer PQ keys over traditional ones, while legacy implementations will safely ignore the unknown algorithm identifier. This approach eases key distribution, as only one key per recipient has to be shared, but provides only transitional authenticity, as the key will have to be replaced before cryptographically relevant quantum computers are introduced.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

TU WIEN Bibliothek Your knowledge hub
Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

CHAPTER 4

Implementation

The designed protocol has been implemented in Golang, in the open-source library go-crypto[1]. In its git repository, the branch `draft-wussler-openpgp-pqc` experimentally implements the latest draft of the specification.

4.1 Dependencies

First and foremost, we investigated support for Kyber, Dilithium and SPHINCS$^+$, to find out whether suitable, secure libraries are already available, or whether development from scratch is necessary.

The CRYSTALS algorithms are provided with a good community-based support, with several viable candidates, among which two were selected:

- crystals-go[2], from Kudelski Security, that provides good support for both algorithms and a well-documented security record.
- circl[3], from Cloudflare, a complementary set to Golang's x/crypto that contains a variety of well-implemented and performant algorithms.

A first implementation used crystals-go, but to reduce the executable size and increase performance it was replaced by circl. This second library has a better designed API, higher throughput, and was already used into the go-crypto library for the X25519, Ed25519, X448, and Ed448 algorithms.

[1] `https://github.com/ProtonMail/go-crypto`
[2] `https://github.com/kudelskisecurity/crystals-go`
[3] `https://github.com/cloudflare/circl`

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

For SPHINCS$^+$ the selection is much more restricted. The only viable candidate is SPHINCSPLUS-golang[4], a straightforward Golang implementation of the round 3 submission recommended by the SPHINCS$^+$ authors on the algorithm's homepage. This library does not offer the same scrutiny as the CRYSTALS counterparts, but is nevertheless valid. Instead of rewriting from scratch, it was decided to rather contribute to the exiting library, especially since the author opted for an MIT licence, a very permissive open source option that easily allows reuse and extension.

4.2 Changes to the Library

4.2.1 Creating a Generic EC Interface

Before implementing the PQ algorithms, a refactoring of the legacy EC code was necessary: a reusable generic interface for ECDH, ECDSA, and EdDSA was created, ready to be reused in the hybrid schemes. Furthermore standalone support for X448 and Ed448 was added, and the library for X25519 and Ed25519 was switched over to circl to minimise the compiled library size and increase performance.

```
type Curve interface {
  GetCurveName() string
}

type ECDSACurve interface {
  Curve
  MarshalIntegerPoint(x, y *big.Int) []byte
  UnmarshalIntegerPoint([]byte) (x, y *big.Int)
  MarshalIntegerSecret(d *big.Int) []byte
  UnmarshalIntegerSecret(d []byte) *big.Int
  MarshalFieldInteger(d *big.Int) []byte
  UnmarshalFieldInteger(d []byte) *big.Int
  GenerateECDSA(rand io.Reader) (x, y, secret *big.Int, err error)
  Sign(rand io.Reader, x, y, d *big.Int, hash []byte)
    (r, s *big.Int, err error)
  Verify(x, y *big.Int, hash []byte, r, s *big.Int) bool
  ValidateECDSA(x, y *big.Int, secret []byte) error
}

type EdDSACurve interface {
  Curve
  MarshalBytePoint(x []byte) []byte
  UnmarshalBytePoint([]byte) (x []byte)
  MarshalByteSecret(d []byte) []byte
  UnmarshalByteSecret(d []byte) []byte
  MarshalSignature(sig []byte) (r, s []byte)
  UnmarshalSignature(r, s []byte) (sig []byte)
  GenerateEdDSA(rand io.Reader) (pub, priv []byte, err error)
  Sign(publicKey, privateKey, message []byte) (sig []byte, err error)
  Verify(publicKey, message, sig []byte) bool
  ValidateEdDSA(publicKey, privateKey []byte) (err error)
```

[4] https://github.com/kasperdi/SPHINCSPLUS-golang

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

```
}

type ECDHCurve interface {
  Curve
  MarshalBytePoint([]byte) (encoded []byte)
  UnmarshalBytePoint(encoded []byte) []byte
  MarshalByteSecret(d []byte) []byte
  UnmarshalByteSecret(d []byte) []byte
  GenerateECDH(rand io.Reader) (point []byte, secret []byte, err error)
  Encaps(rand io.Reader, point []byte)
    (ephemeral, sharedSecret []byte, err error)
  Decaps(ephemeral, secret []byte) (sharedSecret []byte, err error)
  ValidateECDH(public []byte, secret []byte) error
}
```

This interface provides methods to serialize and deserialize points and integers from the native format, generate keys, encapsulate and decapsulate session keys, and validate whether a public key matches its private counterpart.

Each curve implements the corresponding methods of the required interfaces to perform ECDSA, EdDSA, or ECDH. This ensures that curves can be easily interchanged and code does not have to be duplicated when implementing similar PQ algorithms hybrid with different curves.

This procedure did not change the underlying curve implementation for the NIST nor Brainpool curves. It is to be noted, that the latter has never been optimized and therefore suffers from poor performance, presenting a very clear pattern in the tests shown in chapter 5.

4.2.2 Implementing Kyber and Dilithium

The next step has been implementing Kyber in combination with ECDH, trying to minimize code duplication. To hold the algorithm instantiations and the key material a structure pointing to the kyber and curve interfaces has been developed.

```
type PublicKey struct {
  AlgId uint8
  Curve ecc.ECDHCurve
  Kyber kem.Scheme
  PublicKyber kem.PublicKey
  PublicPoint []byte
}

type PrivateKey struct {
  PublicKey
  SecretEC []byte
  SecretKyber kem.PrivateKey
}
```


Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

This allows to easily implement the six different schemes using a unified interface that requires the correct parameters only in the instantiation phase. Three methods are exported: key generation, encryption and decryption.

```
func GenerateKey
  (rand io.Reader, algId uint8, c ecc.ECDHCurve, k kem.Scheme)
  (priv *PrivateKey, err error)

func Encrypt
  (rand io.Reader, pub *PublicKey, msg, publicKeyHash []byte)
  (kEphemeral, ecEphemeral, ciphertext []byte, err error)

func Decrypt
  (priv *PrivateKey, kEphemeral, ecEphemeral, ciphertext, publicKeyHash []byte)
  (msg []byte, err error)
```

When generating a key we instantiate the data structure, requiring the algorithm ID and the corresponding EC and Kyber instances. Then, public and private keys can be used with the `Encrypt` and `Decrypt` methods without having to specify the primitives.

Both `Encrypt` and `Decrypt` use a shared method to perform the KDF procedure, for which `KMAC256` had to be implemented starting from the `cSHAKE` primitive.

Since no PQ signature algorithm was yet implemented, to test Kyber encryption keys as standalone a Kyber encryption subkey was attached to a plain Ed25519 primary key.

Then, hybrid dilithium support was implemented using two separate internal representations, one for EdDSA and one for ECDSA. In fact, the two algorithms differ significantly in the data encoding and use two different underlying implementations to handle the curve component.

The EdDSA data structure is as follows:

```
type PublicKey struct {
  AlgId uint8
  Curve ecc.EdDSACurve
  Dilithium dilithium.Mode
  PublicPoint []byte
  PublicDilithium dilithium.PublicKey
}

type PrivateKey struct {
  PublicKey
  SecretEC []byte
  SecretDilithium dilithium.PrivateKey
}

func GenerateKey
  (rand io.Reader, algId uint8, c ecc.EdDSACurve, d dilithium.Mode)
  (priv *PrivateKey, err error)
```


Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

While the ECDSA stores the points by their integer coordinates:

```
type PublicKey struct {
  AlgId uint8
  Curve ecc.ECDSACurve
  Dilithium dilithium.Mode
  X, Y *big.Int
  PublicDilithium dilithium.PublicKey
}

type PrivateKey struct {
  PublicKey
  SecretEC *big.Int
  SecretDilithium dilithium.PrivateKey
}

func GenerateKey
   (rand io.Reader, algId uint8, c ecc.ECDSACurve, d dilithium.Mode)
   (priv *PrivateKey, err error)
```

Both EdDSA and ECDSA offer a similar signature and verification interface:

```
func Sign(priv *PrivateKey, message []byte) (dSig, ecSig []byte, err error)
func Verify(pub *PublicKey, message, dSig, ecSig []byte) bool
```

Finally, all keys offer a `Validate` method to prevent KOpenPGP [BHP22] attacks. This validates whether the public key component matches the one derived from the private key, preventing key replacement attacks:

```
func Validate(priv *PrivateKey) (err error)
```

4.2.3 Implementing SPHINCS$^+$

Being SPHINCS$^+$ a standalone algorithm, not used in a hybrid configuration, its implementation was limited to a shallow wrapper around the SPHINCSPLUS-golang library, mapping the different security parameters to the library instantiation.

The data structures are defined as follows:

```
type PublicKey struct {
  ParameterSetId ParameterSetId
  Mode Mode
  Parameters *parameters.Parameters
  PublicData *sphincs.SPHINCS_PK
}

type PrivateKey struct {
  PublicKey
  SecretData *sphincs.SPHINCS_SK
}
```


Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Here `ParameterSetId` corresponds to the OpenPGP configuration option that is then mapped to the `Parameters` of the underlying library.

4.2.4 Additional Changes Required for Integration

In order to integrate the library some additional pivotal changes were necessary:

- A new encoding type `OctetArray` has been introduced. As previously discussed in section 3.4 we opted to use fixed length binary strings, that were not previously available in OpenPGP.
- The `Config` structure, used to pass key generation options, had to be altered to take a `SphincsPlusParameterId` option, that allows customization when generating SPHINCS$^+$ keys.
- The algorithm to tweak subkey preference has been changed to favour PQ keys over traditional when looking for an encryption key candidate.
- Version checks were implemented across the codebase to ensure the version binding is enforced.
- Unit, integration, and end-to-end tests were added to the library for the new algorithms.

4.2.5 Design Changes During the Implementation Phase

The first draft, version 00, was published in December 2022, providing a permanent link to add references to the specifications in the code, making procedures well documented. The second draft, version 01, was published in March 2023 and is the one that this thesis is based on. It adapted the text to the latest changes in version 08 of the OpenPGP draft specification [WHWY23].

In this section we describe some of the changes that were done after having started the implementation, that led to the changes across the draft versions, gathering feedback from development process.

Kyber Parametrization

It was then attempted to parametrize the Kyber and Dilithium components, similarly to the choice taken for SPHINCS$^+$. For instance, this would have meant having Kyber + X25519 as algorithm ID 29 featuring a parameter in the packet specifying which security level of Kyber to use. This option was then excluded because it added complexity to the protocol without a real advantage in crypto agility. It was valued more important to have fixed and clear combinations, than to allow users to pick which PQ security level was to be used with different curves. Reasonable combinations are provided by design, that ensure a similar expected security level between curves and PQ algorithms.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Lower Security Parameters

In an early version, the algorithm selection included also Kyber512 and Dilithium2, but we opted to remove it preferring only NIST security level 3 and 5 candidates: Kyber768, Kyber1024, Dilithium3, and Dilithium5. This was done because aggressive bandwidth or performance optimizations are not necessary in OpenPGP, rather a conservative approach with respect to security is preferred. This was also confirmed with the community when gathering feedback at IETF 115 in London, and corresponds with the recommendation from the NSA [Age22]. The same feedback also influenced the choice of EC for both signatures and PKE changing respectively from NIST-P-384 and NIST-P-521 to NIST-P-256 and NIST-P-384, as well as brainpoolP384r1 and brainpoolP512r1 being respectively downgraded to brainpoolP256r1 and brainpoolP384r1.

Key Combiner

The Kyber + ECDH key combiner has been changed due to security concerns, as it could collide with the existing ECDH key combiner. The new revision is using KMAC256 instead of plain SHA-3, introducing domain separation from the existing ECDH KDF. Furthermore, in version 01, the KDF construction has changed to be IND–CCA2 secure.

Preference of PQ Keys

When having to develop an algorithm to select candidate encryption subkeys, we opted to prefer PQ over traditional keys, a measure necessary to allow a smooth migration for PQ capable implementations.

Binding PQ to Version 6

Given the security benefits of version 6 packets, we decided to bind the PQ algorithms to this version for PKESK, signature, and key packets. Part of these features were not yet fully implemented in go-crypto, therefore benchmarking incurred in some delays.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

CHAPTER 5

Results and Performance Analysis

To measure the performance of the algorithms and implementation we timed end-to-end key generation, parsing, encryption, signing, and verification.

Our objective with these measurement is to provide a realistic user perspective and reason about the user experience when using these algorithms. We therefore chose to measure the end-to-end performance of the implementation rather than measuring the performance of the algorithm itself, that has generally been well documented in the context of the NIST PQ competition [MAA+22].

In order to evaluate the benchmark results we need to look into the structure of the generated version 6 certificates. Here we use a very plain structure, that is also the default for go-crypto, with a primary certification and signature capable public-private keypair and an encryption capable subkey. The primary key signs:

- a direct key signature containing the user preferences,
- a single primary user identity,
- the encryption subkey via a binding signature.

Given the key structure:

- Key generation operation consists of two different primitive key generations and three signature generations.
- Key parsing consists of three signature verifications.
- Encryption, decryption, signing and verification are instead just the primitive operations over 1 KB of randomly generated plaintext.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

In these tests, we compare the newly proposed algorithms with the the matching set of existing algorithms: RSA-2048, RSA-3072, RSA-4096, Ed25519, Ed448, P256, P384, Brainpool256, and Brainpool384. The matching between signing and KEM algorithms is shown in table 5.1.

Primary key Algorithm	Encryption Sub-key Algorithm
RSA	RSA
Ed25519	X25519
Ed448	X448
ECDSA-NIST-P-256	ECDH-NIST-P-256
ECDSA-NIST-P-384	ECDH-NIST-P-384
ECDSA-brainpoolP256r1	ECDH-brainpoolP256r1
ECDSA-brainpoolP384r1	ECDH-brainpoolP384r1
Dilithium3 + Ed25519	Kyber768 + X25519
Dilithium5 + Ed448	Kyber1024 + X448
Dilithium3 + ECDSA-NIST-P-256	Kyber768 + ECDH-NIST-P-256
Dilithium5 + ECDSA-NIST-P-384	Kyber1024 + ECDH-NIST-P-384
Dilithium3 + ECDSA-brainpoolP256r1	Kyber768 + ECDH-brainpoolP256r1
Dilithium5 + ECDSA-brainpoolP384r1	Kyber1024 + ECDH-brainpoolP384r1
SPHINCS+–simple–SHA2	Kyber1024 + X448
SPHINCS+–simple–SHAKE	Kyber1024 + X448

Table 5.1: Default matching for public and private keys in go-crypto

5.1 Artifact Size

We start by analyzing the artifact size for each considered algorithm combination and operation in fig. 5.1. These results are implementation specific, and as much as they do not vary depending on platform, other implementations might serialize packets differently, effectively generating similar but not identical sizes. As it is here done for performance benchmarking, we use the default settings and 1 KB uncompressed plaintext to generate the results.

At a first glance it appears evident that PQ algorithms, and in particular SPHINCS$^+$ have much larger artifacts than any traditional algorithm, 4096 bit RSA included. No PQ algorithm in the standardization process unfortunately provides artifacts under a few KB in size, let alone in the hundred of bytes like EC. As discussed in chapter 2, this

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

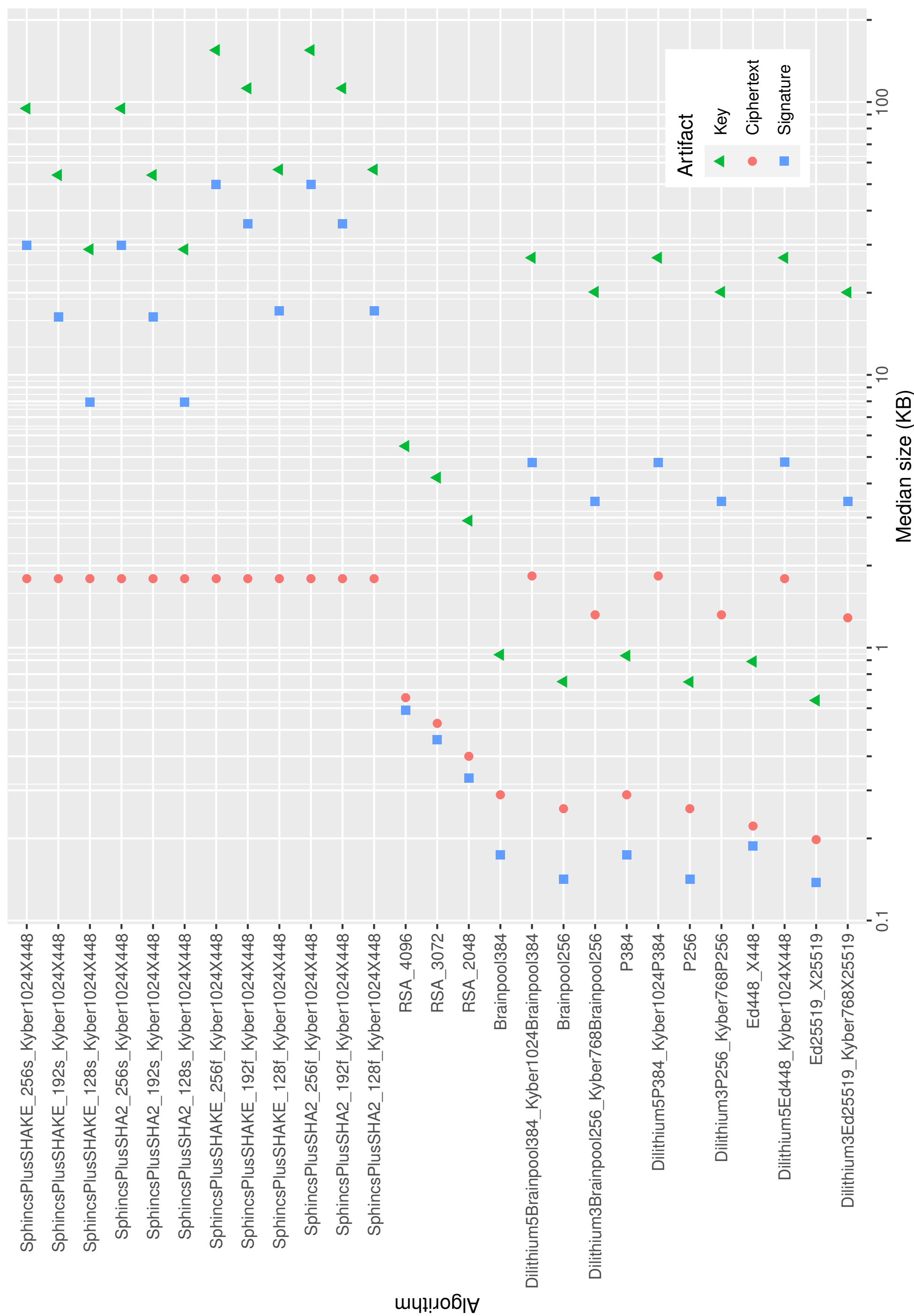


Figure 5.1: Overview of the artifact size for all newly implemented and reference algorithms for 1 KB of data. The plot is logarithmic in the size of the artifact.

tradeoff is well-known, and widely accepted in the community, and part of this project's contribution is to investigate what impact this has on the existing protocol.

It is to be noted, that for encryption and signature the sizes reported in fig. 5.1 refer only to the algorithm overhead, i.e. excluding the signature plaintext and the encrypted ciphertext. For instance, for a 1KB large email, X25519 will add 0.2KB of overhead totalling 1.2KB message size, while Kyber768 combined with X25519 will add 2KB of overhead, totalling 3KB.

Fortunately for OpenPGP, this does not seem to pose a problem, except for some edge-cases such as cleartext signed messages where a base64 encoded signature is appended to a human-readable message, they will appear to users as large chunks of unreadable data eclipsing the actual message.

Other issues might appear only at scale, when handling large storage of messages or signatures; for instance migrating billions of signatures from 138 bytes for Ed25519 to 3432 bytes for Ed25519 + Dilithium3 can be expensive. Given that around 3.5KB is the smallest signature offered, this cost has to be taken in consideration in the process of PQ migration. Ciphertexts also present the same problem: for 1 KB of plaintext there will be a 129% overhead.

5.2 Desktop Performance Analysis Methodology

For desktop devices, the measurements were taken using the integrated Golang benchmarking tool. For each measurement we repeated the operation 8 times and averaged the time, repeating this procedure 10 times.

Measurements were taken on an idling x86 Intel(R) Core(TM) i5-8265U CPU, clocked at 1.60 GHz. A consumer laptop CPU was chosen to ensure realistic measurements from the user perspective.

We can first observe an overview of the median performance of all considered algorithms in fig. 5.2. For this first plot, we selected a logarithmic scale on the time axis: performance varies in a very large range, from 0.15 ms to 15 s, given the tradeoffs between security, size, and speed.

In this plot, it can be seen that, except for SPHINCS$^+$, all operations remain into the same order of magnitude, and that the lattice-based schemes preserve the computational cost of the elliptic curves, also in the hybrid setting. In the following sections we will break down the analysis by operation and algorithms, with detailed considerations about each comparable set using linear graphs.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

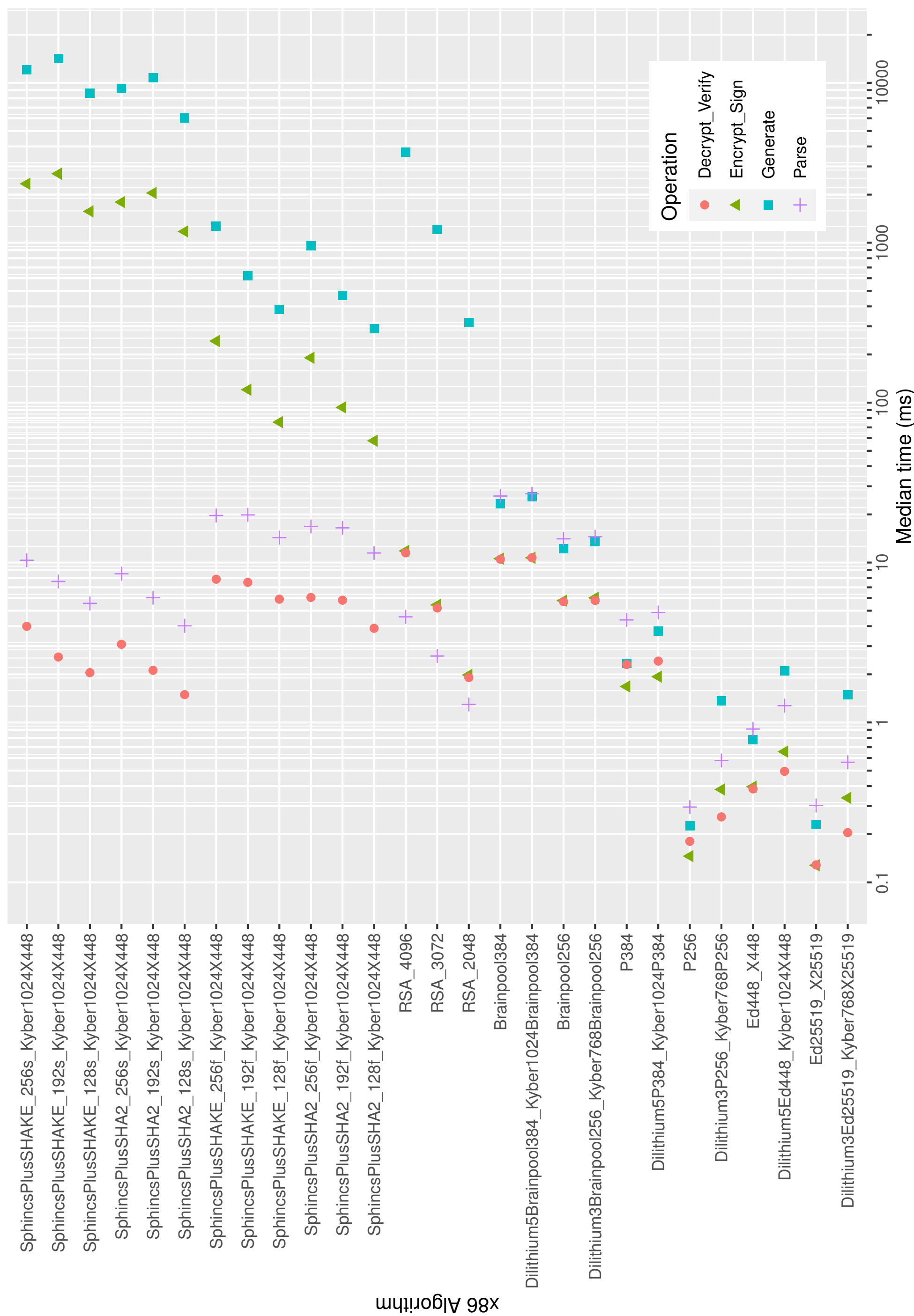


Figure 5.2: Overview of the x86 performance for all newly implemented and reference algorithms for 1 KB of data. The plot is logarithmic in the operation time.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Figure 5.3: Overview of the ARM performance for all newly implemented and reference algorithms for 1 KB of data. The plot is logarithmic in the operation time.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

5.3 Mobile Performance Analysis Methodology

In order to perform measurements on mobile devices, we first had to compile GopenPGP[1] using go-mobile[2] for Android: this builds a library that can be accessed from applications via bindings. A simple application using Google's instrumented test suite `AndroidJUnit4` was then developed and run on a Fairphone 4, featuring an 8-core 64-bit Qualcomm LAGOON ARM CPU clocked at 2.1 GHz. This application ran each test 16 times and provided the raw execution time. Similarly to the desktop tests, the results are plotted using a logarithmic scale for the time axis, in fig. 5.3.

It can be observed that the distribution of the results is very close to the x86 performance presented in fig. 5.2. Some overhead for the fastest operations can be seen, due to the language bindings, with most of the curve operations on ARM taking slightly more than 1 ms instead of just under.

In the case of SPHINCS$^+$ performance is mostly dependent on the underlying hash implementation, and since both platforms use highly optimized assembly code the performance pattern is similar.

5.4 Key Generation

The benchmarked OpenPGP key generation consists in generating two keys and three signatures, making it the slowest operation for most algorithms.

We start by comparing the two computationally most expensive traditional and PQ algorithm, RSA and the compact variant of SPHINCS$^+$, shown for both architectures in fig. 5.4.

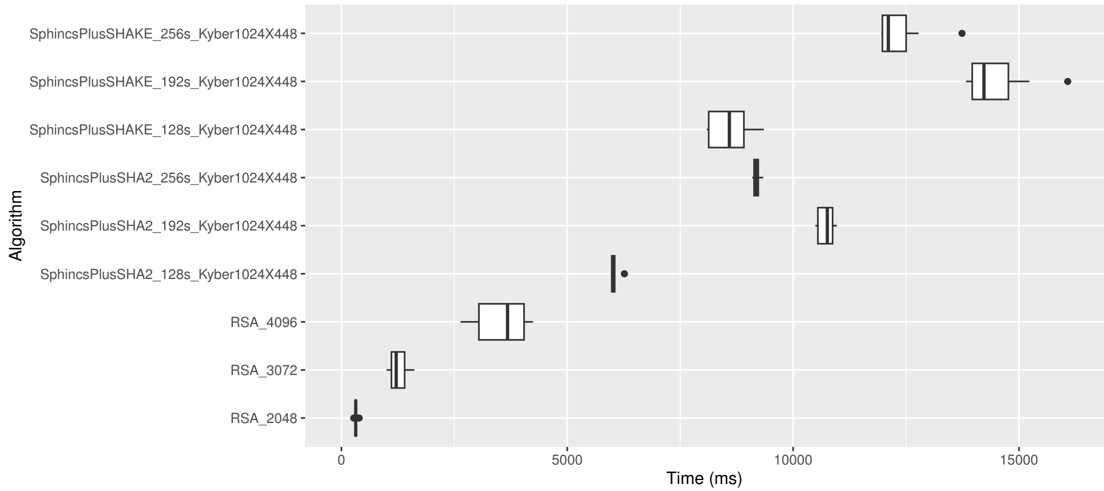
While both algorithms require a long time to generate keys, RSA is slow because of the probabilistic prime search, while SPHINCS$^+$ is slow because of signature generation. This performance difference will be evident later on when comparing signature creation time. It is also interesting to note that RSA seems to have an unoptimized prime search on ARM architectures, while SPHINCS$^+$ efficiently runs on both. This results on significantly different times for x86 key generation, but comparable for ARM.

In fig. 5.5 we can observe the fast version of SPHINCS$^+$ versus RSA. From the data we can infer that the performance of the fast variant of SPHINCS$^+$ is faster than the currently deployed RSA algorithms, with its slowest version, being as fast today's common RSA default, 3072 bits.

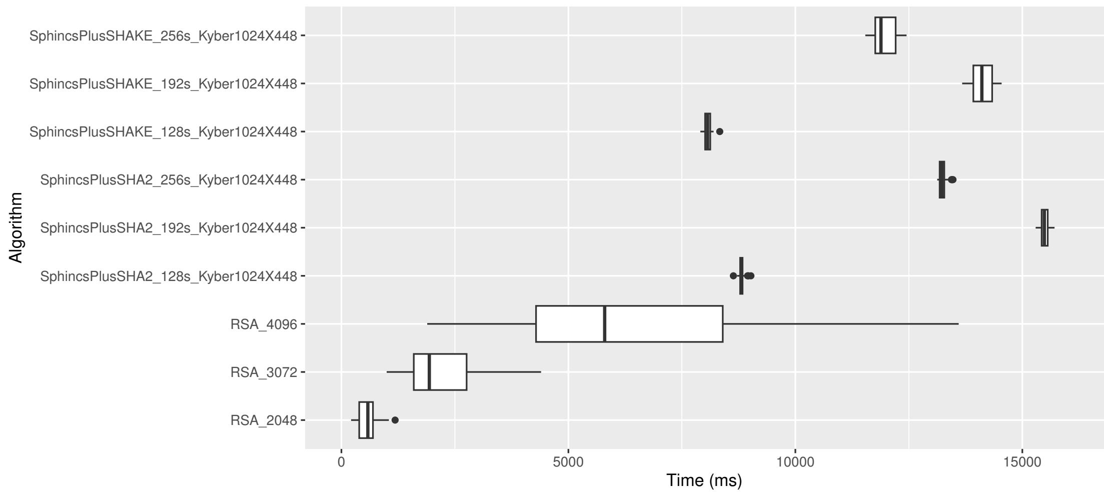
Since key generation is generally a one-off operation, this long processing time is generally considered acceptable from a user experience point of view. The main compromise here is key size: in fig. 5.1 we can observe how RSA has a key size between 2 and 4 KB, while SPHINCS$^+$ keys lie in the range from 20 to 200 KB.

[1] `https://github.com/ProtonMail/gopenpgp`
[2] `https://github.com/golang/mobile`

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub



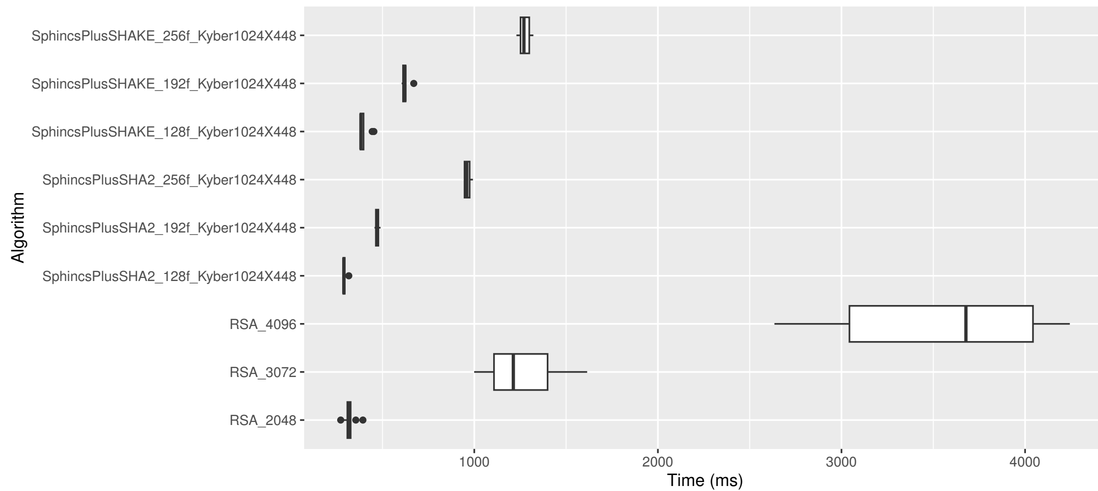
(a) x86 implementation



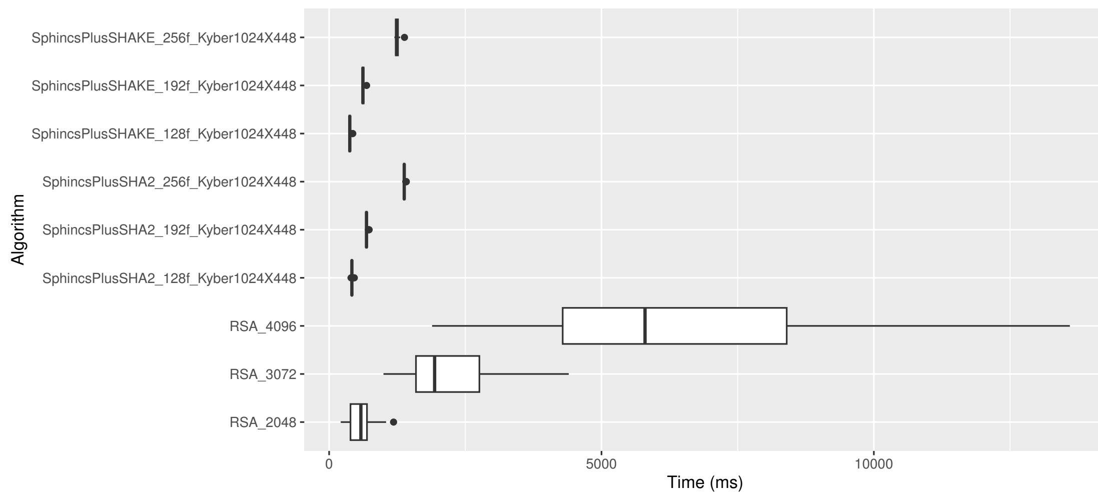
(b) ARM implementation

Figure 5.4: Time comparison for key generation of compact SPHINCS$^+$ vs RSA

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub



(a) x86 implementation

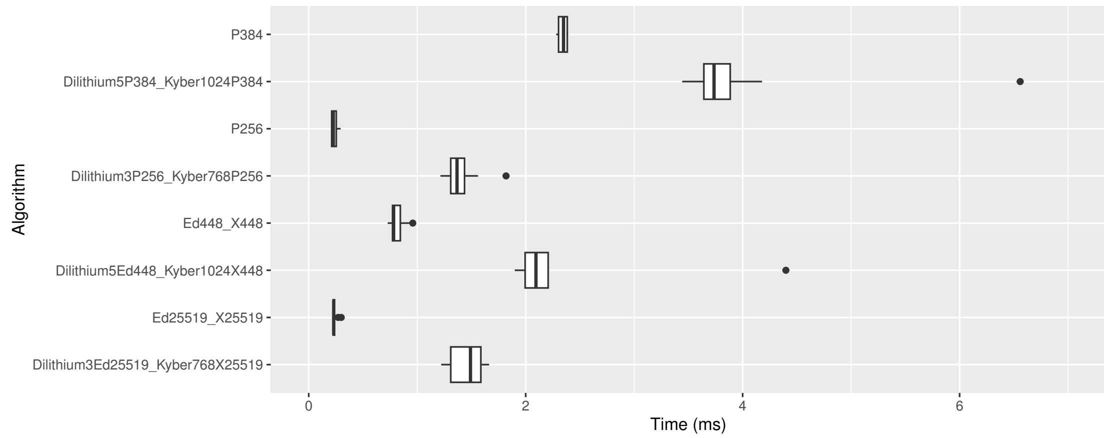


(b) ARM implementation

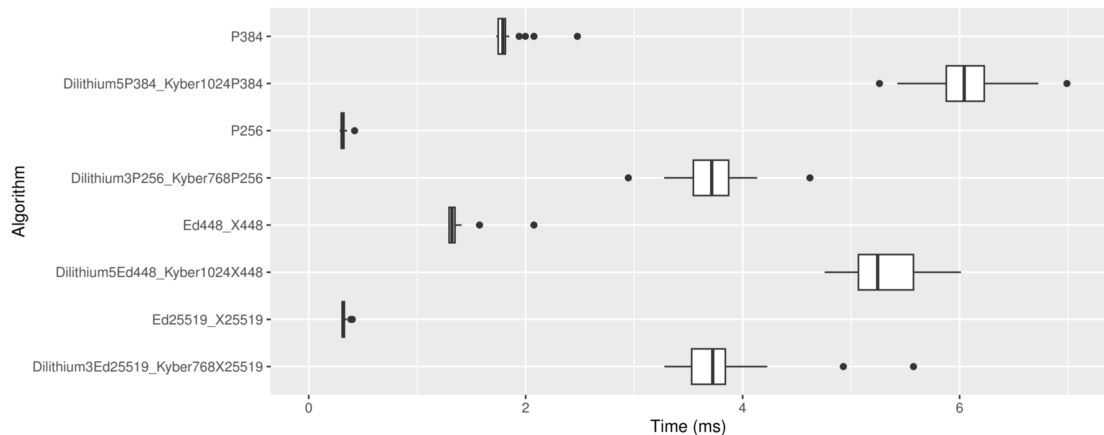
Figure 5.5: Time comparison for key generation of fast SPHINCS$^+$ vs RSA

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

For a completely different speed range we can now consider EC performance, shown in fig. 5.6 and fig. 5.7. The two charts were split because of the significant performance penalty of the Brainpool curves.



(a) x86 implementation



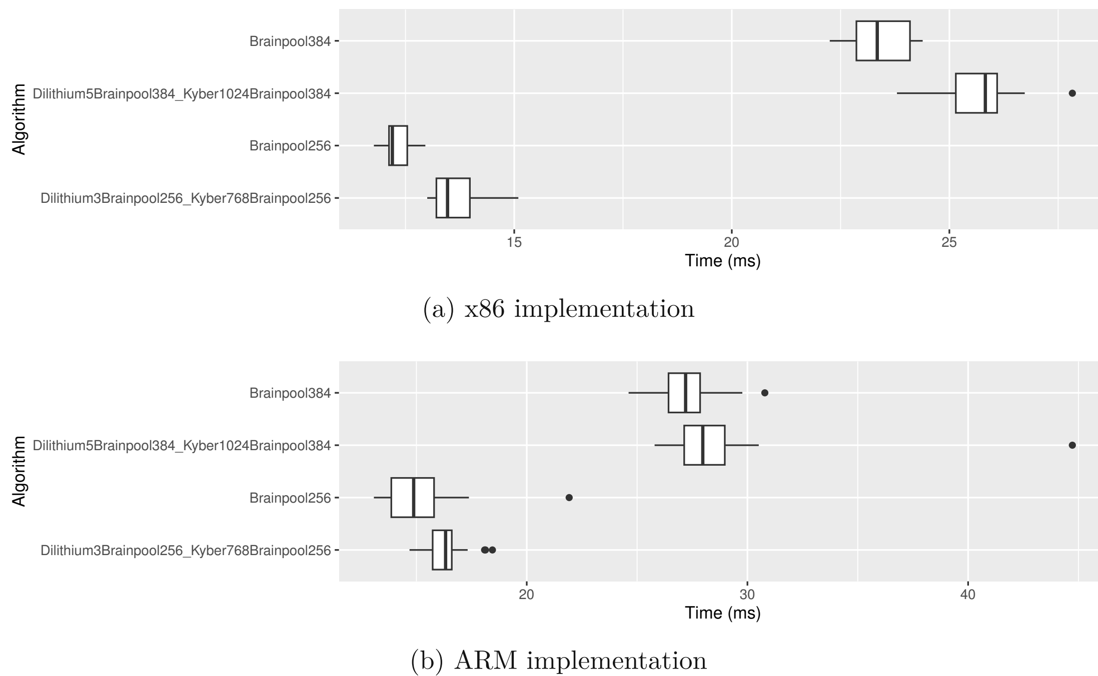
(b) ARM implementation

Figure 5.6: Time comparison for key generation of Dilithium vs NIST and CFRG curves

In these plots we compare plain elliptic curves versus hybrid operation, noting that the plain and hybrid implementations share the same code for the EC component. A constant overhead of approximately 2 ms on x86 and 4 ms on ARM can be reliably seen, therefore hinting to some missing instructions in the ARM set having a negative impact on performance.

In both cases the overall impact is very small, since the operations stay in the millisecond range, even though for Ed25519 this represents a 10-fold increase.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub



(a) x86 implementation

(b) ARM implementation

Figure 5.7: Time comparison for key generation of Dilithium vs Brainpool curves

5.5 Key Parsing

Once a key has been generated and shared, we need to consider the performance of parsing it on the other end. In OpenPGP this consists not only of loading the data from disk into the correct data structure, but also of verifying all the embedded signatures.

Since signature verification is a fast operation for all considered algorithms, we were able to collect all the information in a single plot in fig. 5.8.

It can be seen that apart from the poor implementation of Brainpool in go-crypto, all EC algorithms lie under the 10 ms threshold. For SPHINCS$^+$ it is interesting to note that the fast variant is twice as slow in signature verification compared to the compact one. This phenomena happens because the algorithm is optimized for speed in signature generation rather than verification. Said this, the operation is still in the 5 to 25 ms range and poses no problematic restriction to SPHINCS$^+$ operations.

It is interesting to note that while on x86 architectures the SHA–2 variant of SPHINCS$^+$ is faster than SHAKE, for ARM the converse is true. The other algorithms, even if slower on ARM, present the same patterns on both architectures.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

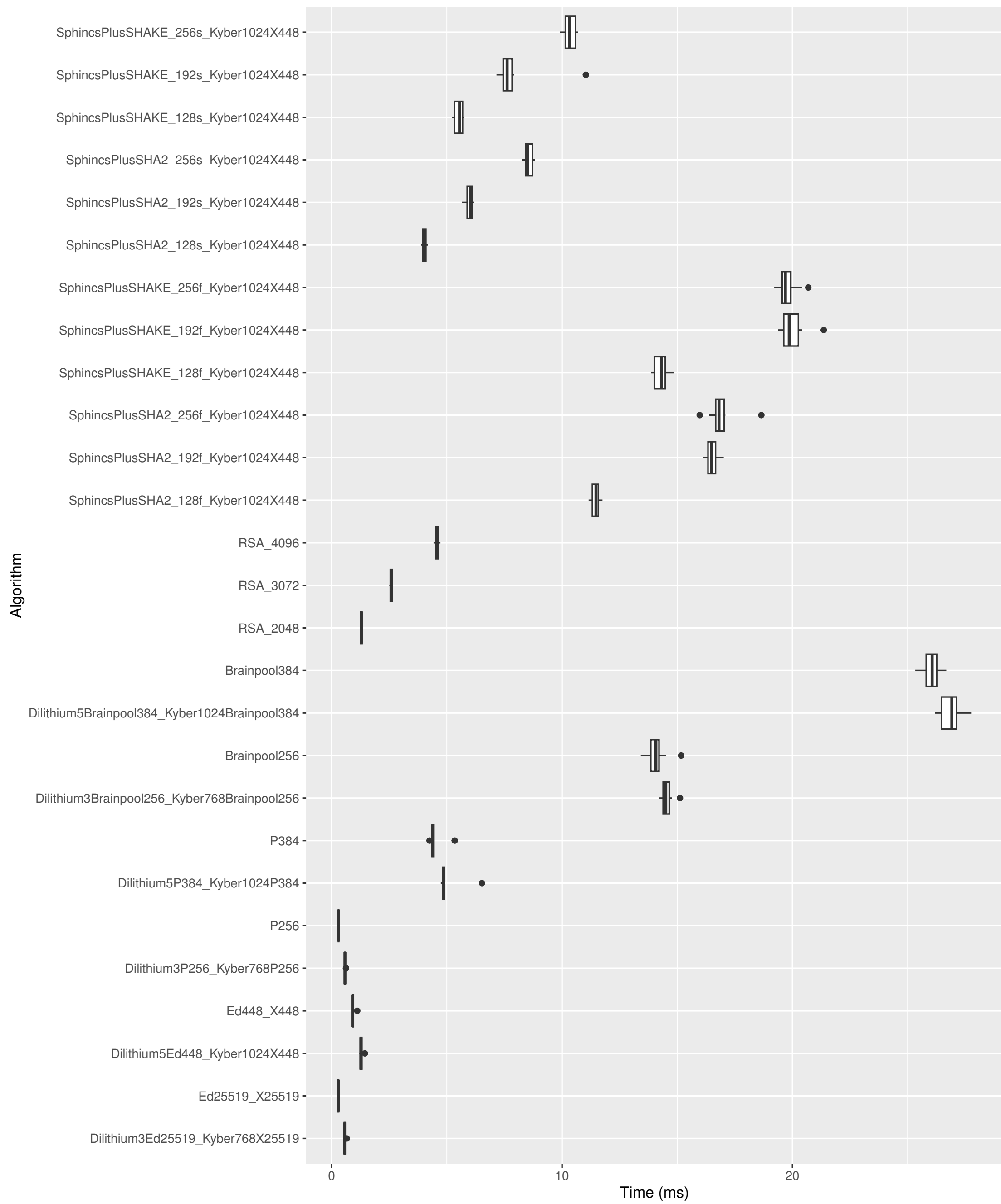


Figure 5.8: Time comparison of key parsing on x86

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

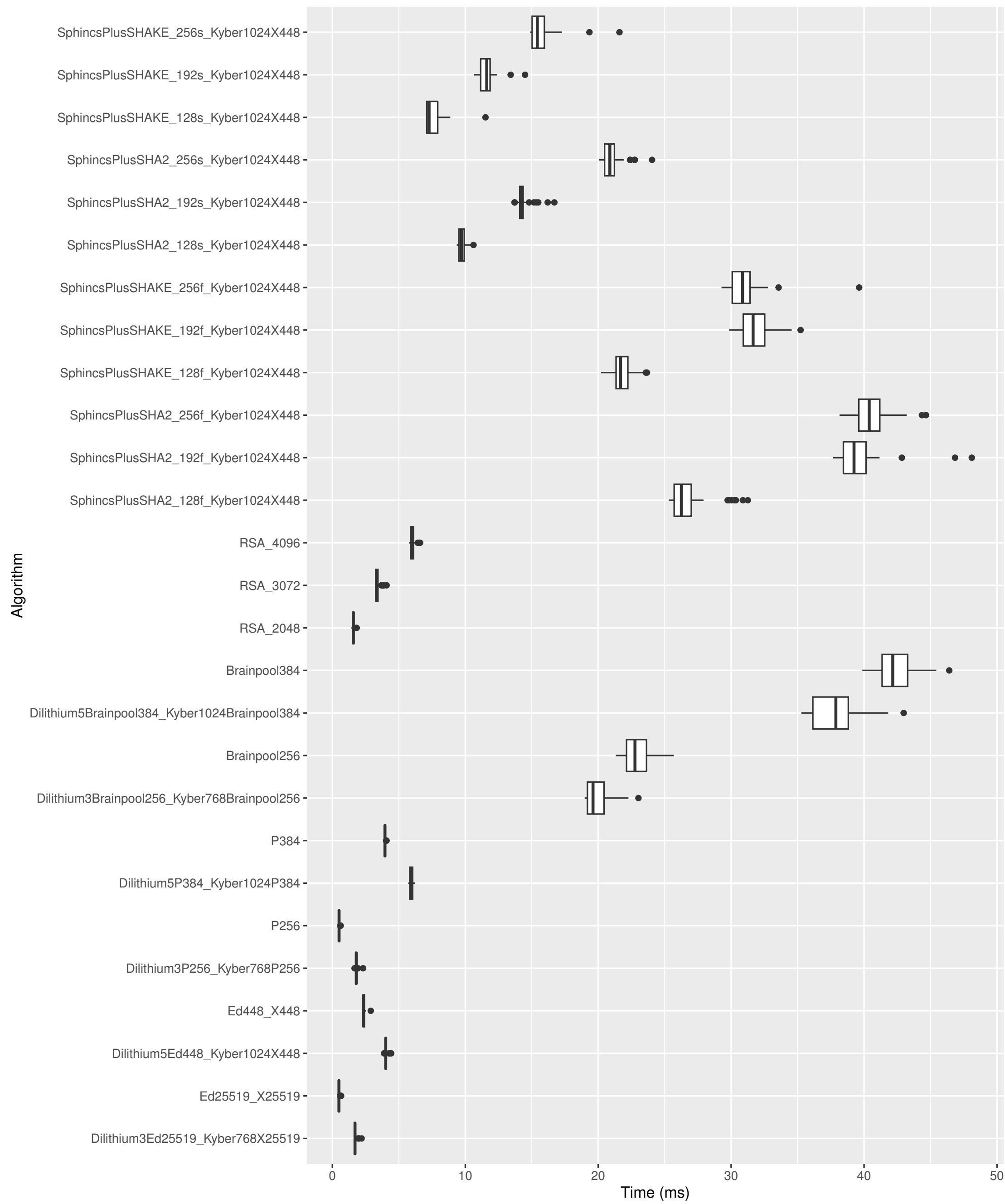
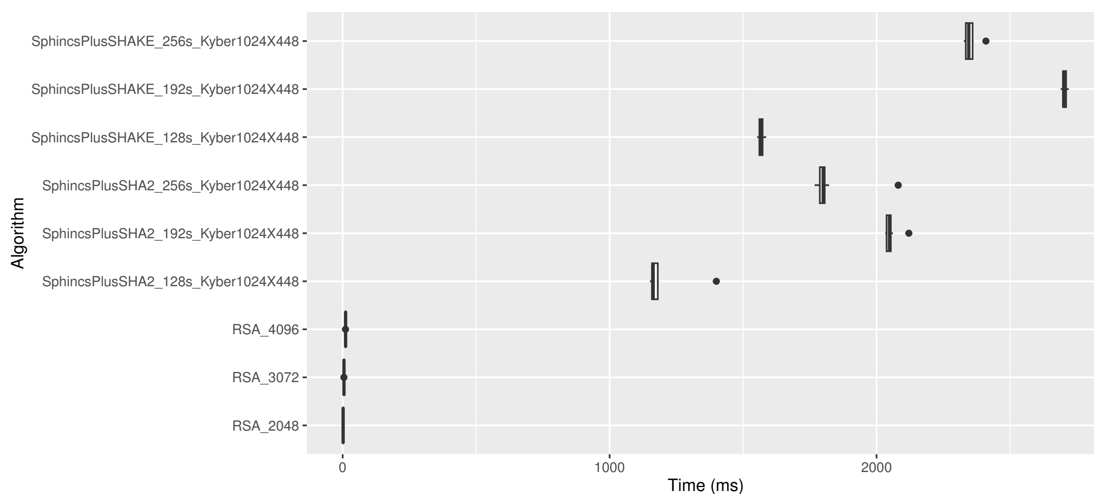


Figure 5.9: Time comparison of key parsing on ARM

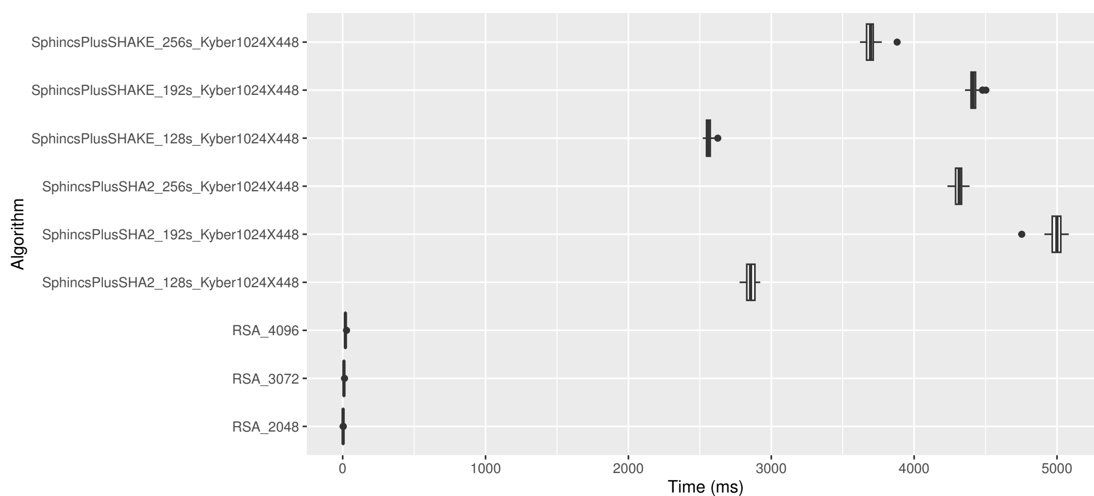
Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

5.6 Signature Generation

When signing data, SPHINCS$^+$ has a considerable speed and artifact size tradeoff. As it can be seen in fig. 5.10, the compact variant up to 3 s on x86 and up to 5 s on ARM architectures. The fast variant, displayed in fig. 5.11 requires instead up to 250 ms on x86 and up to 500 ms on ARM, still significantly slower than any other algorithm deployed today. Here developers will have to carefully consider whether their application is suitable for SPHINCS$^+$ and account for the resulting user experience.



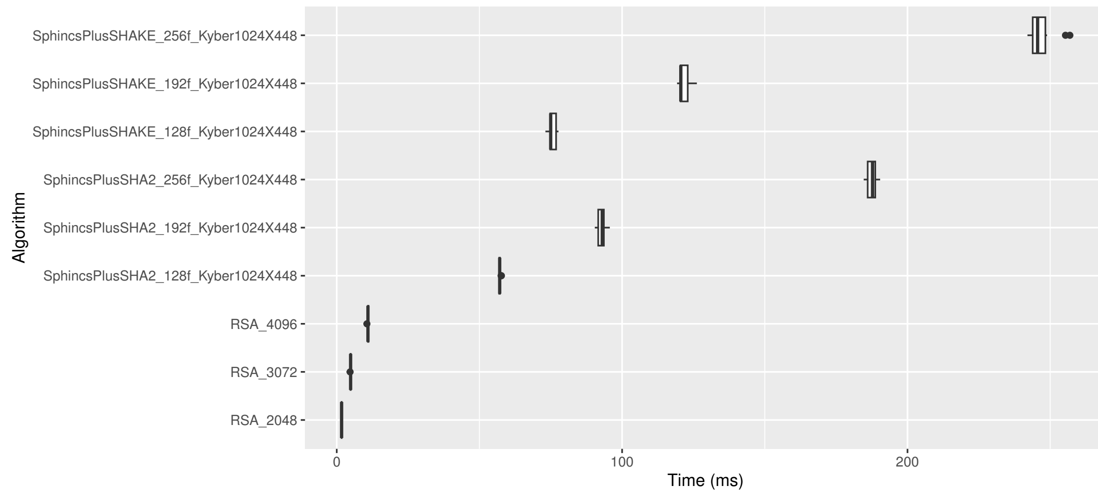
(a) x86 implementation



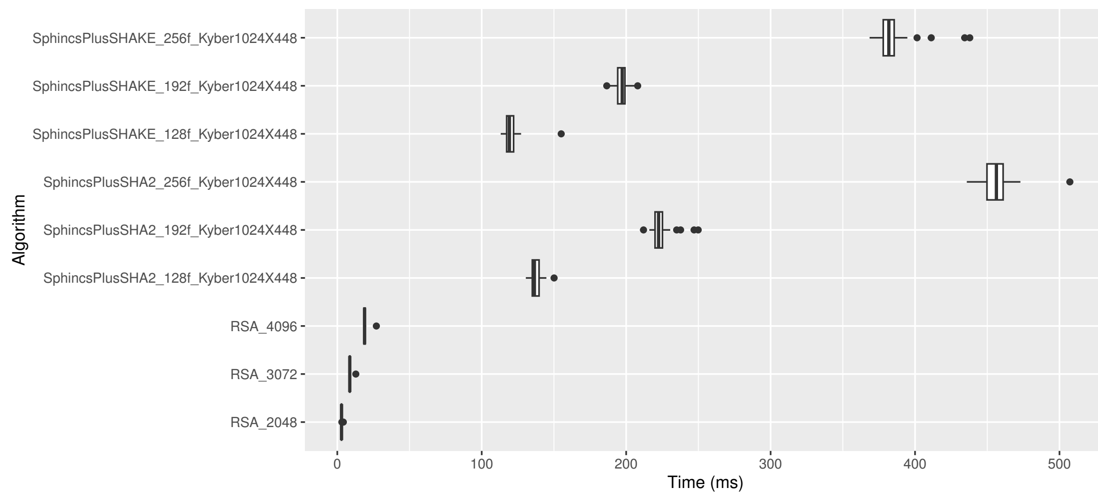
(b) ARM implementation

Figure 5.10: Time comparison for signature generation of the compact variant of SPHINCS$^+$ vs RSA

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub



(a) x86 implementation



(b) ARM implementation

Figure 5.11: Time comparison for signature generation of the fast variant of SPHINCS$^+$ vs RSA

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

In fact, both variants rank worse than RSA by 1 or 2 orders of magnitude, and will most likely imply a user experience change when implemented in applications. Developers will need to consider progress indicators or signalling, especially when using the compact version. For some uses this tradeoff is acceptable, for instance software signing, where 3 seconds to produce a signature might be acceptable as long as verification is fast.

With respect to size, SPHINCS$^+$ signatures start at 7.9 KB, forcing developers to hide them from users, for instance in the already mentioned example of the cleartext signed messages.

For elliptic curves, like key generation, we have a substantial difference between hash and lattice based schemes, as seen in fig. 5.12 and fig. 5.13.

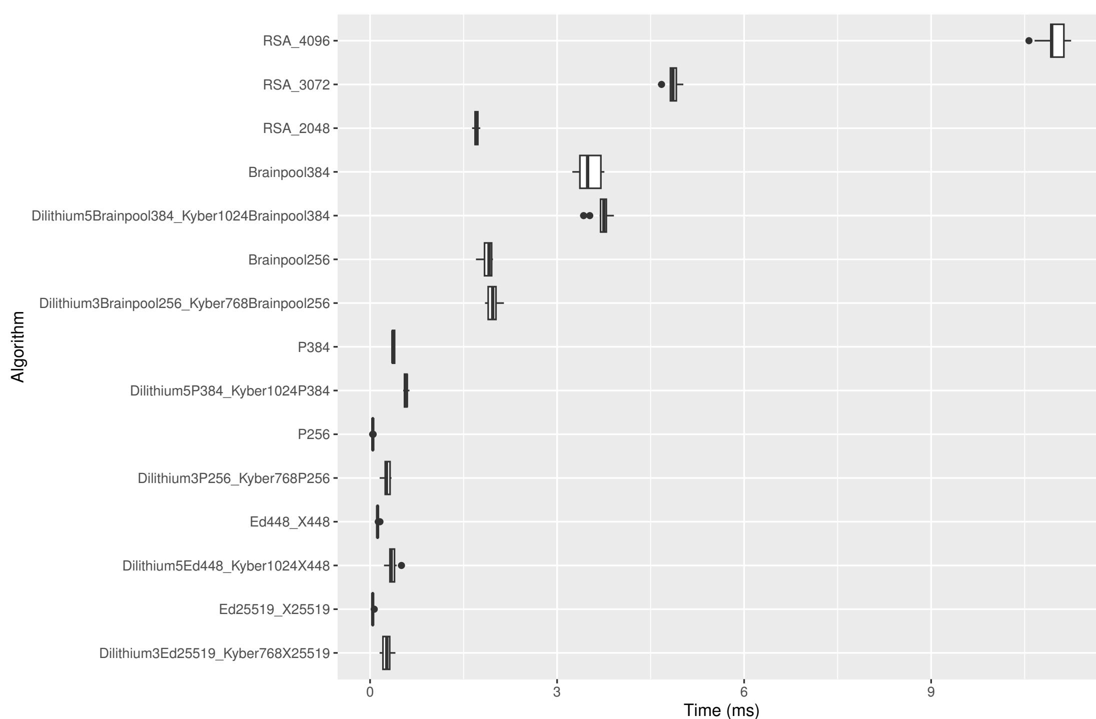


Figure 5.12: Time comparison of signature generation on x86 for Dilithium vs EC

We can observe that except Brainpool curves, all hybrid algorithms are all well under 2 ms, thus faster than 2048 bit RSA, that already provides a reasonably fast performance. There is also no significant difference between the two architectures, with ARM being slower but presenting the same patterns as x86.

For Dilithium, the only tradeoff with respect to the traditional algorithms is artifact size: signatures shift from approximately 100 bytes to over 3 KB, as it can be seen in fig. 5.1.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

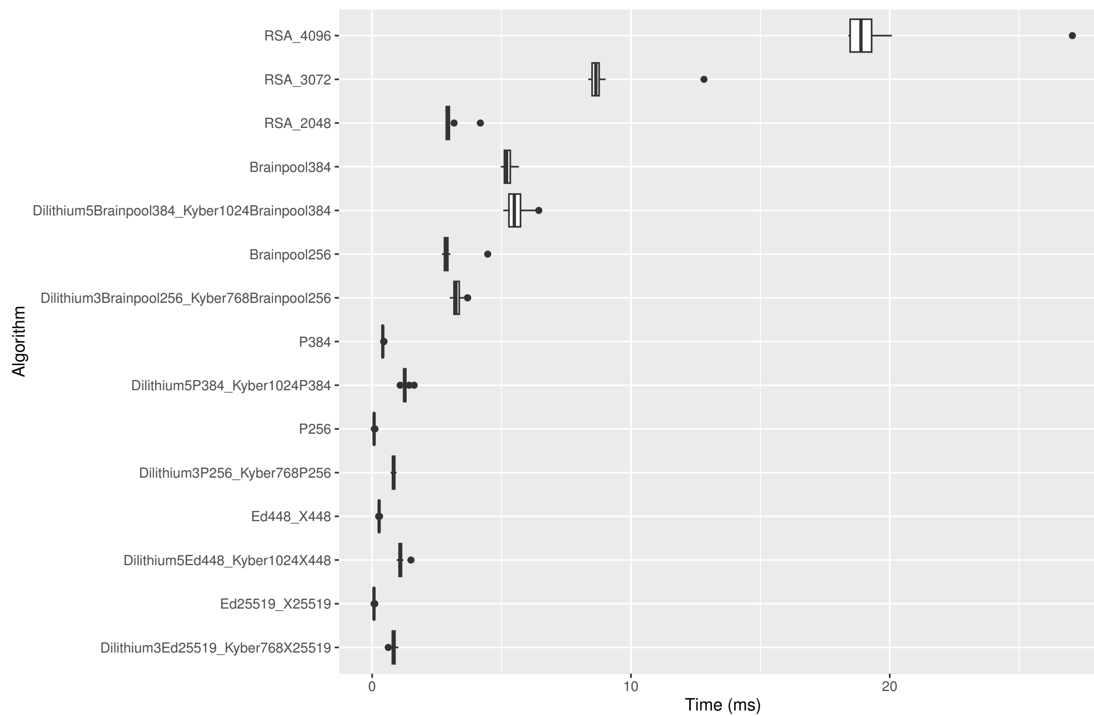


Figure 5.13: Time comparison of signature generation on ARM for Dilithium vs EC

5.7 Signature Verification

In fig. 5.14 and fig. 5.15 it can be seen how signature verification lies into the same range under 15 ms per operation for all algorithms, SPHINCS$^+$ included. Signature verification, shows similar results to key parsing, that mostly consists of 3 verification operations, therefore the same observations apply also here.

A direct consequence of this is that SPHINCS$^+$ is expensive when generating keys or signatures, but not when verifying: the user picking those algorithms is the one that has to pay their cost, while others can efficiently consume the artifacts.

5.8 Encryption

In the case of encryption we have only the lattice-based candidate Kyber, that provides an extremely small overhead of less than 0.2 ms with respect to the pure EC operations, as shown in fig. 5.16. This holds even for the fastest curve in our dataset, X25519, that in the hybrid mode encrypts in 0.11 ms versus 0.08 ms when used as standalone on x86. For ciphertexts the only relevant difference is the artifact size, which overhead increases 10-fold.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

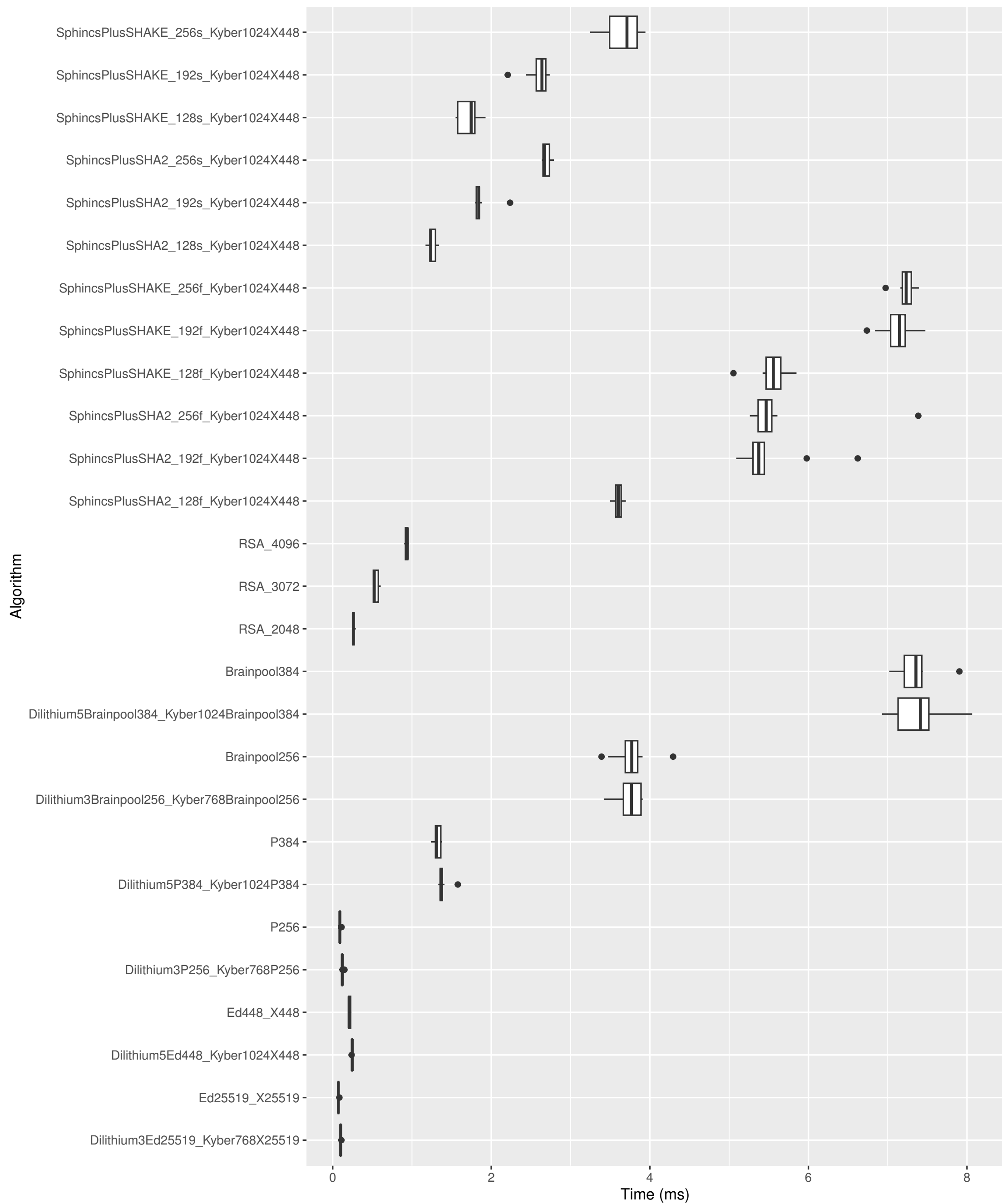


Figure 5.14: Time comparison of verification on x86

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

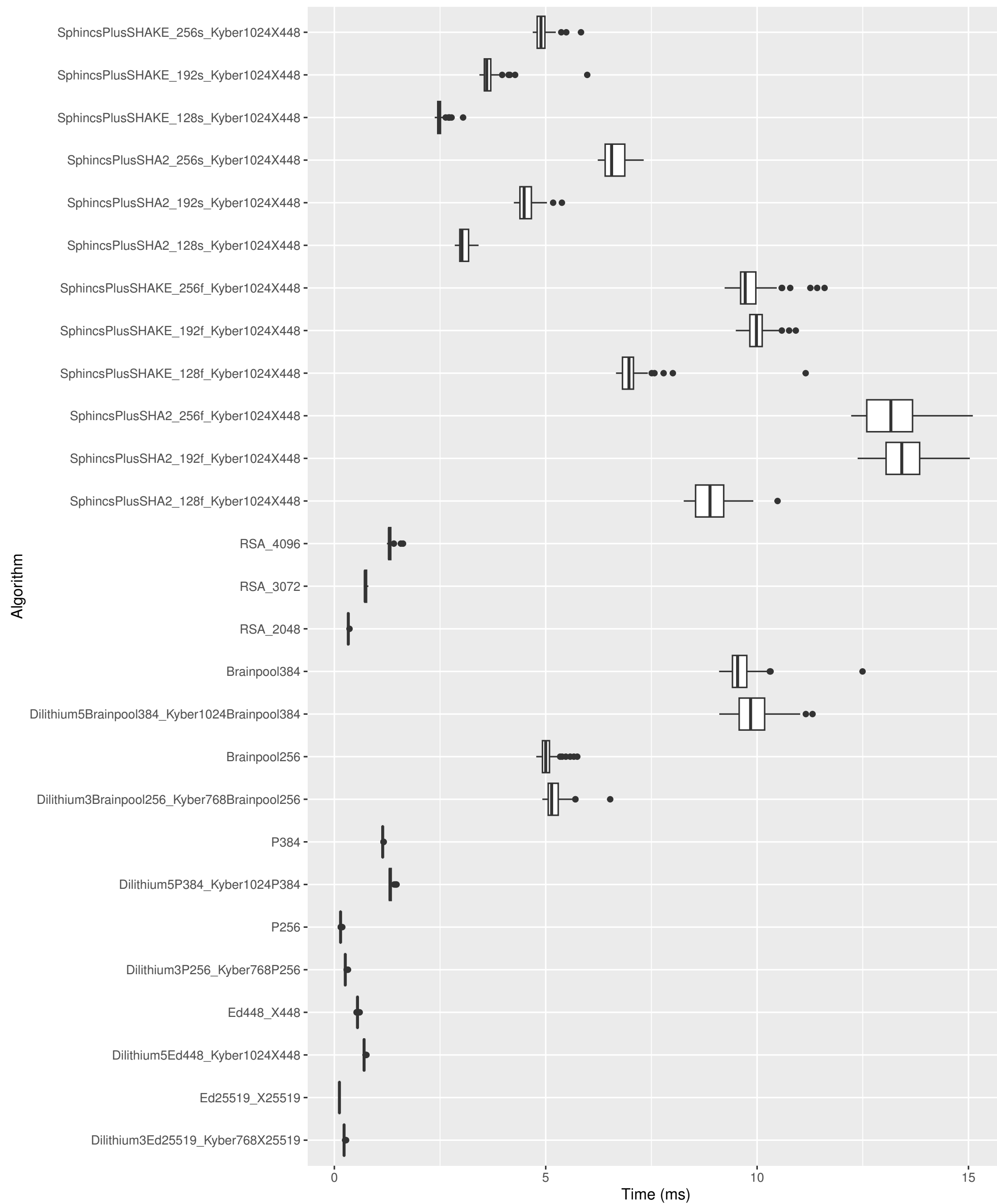
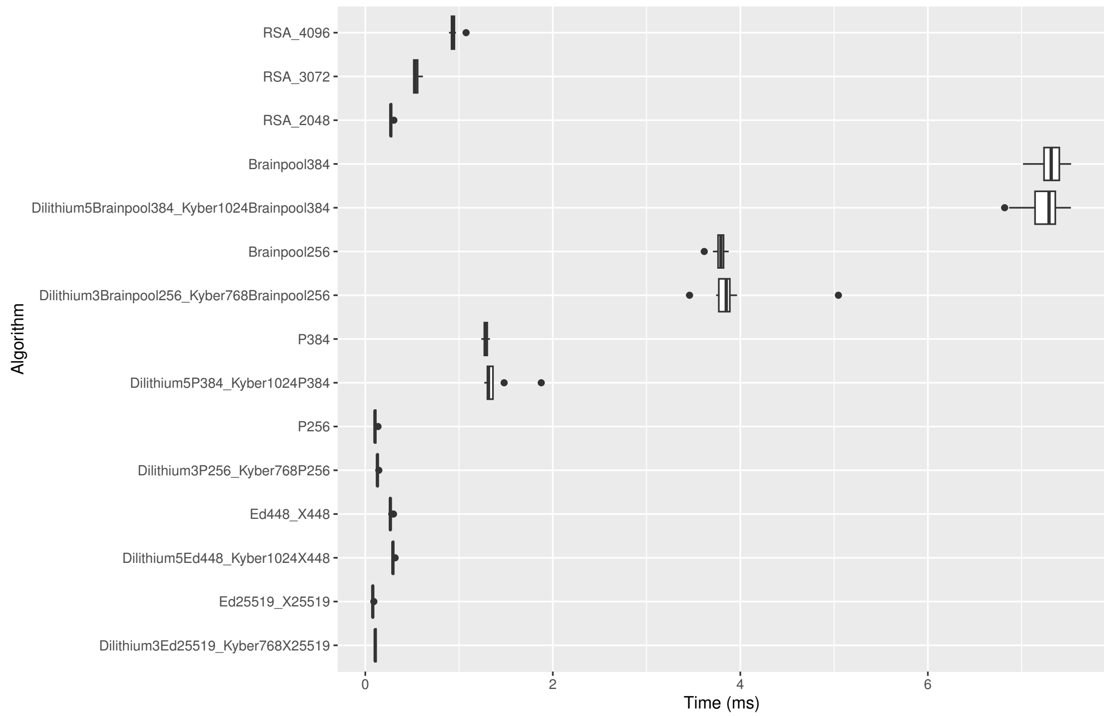
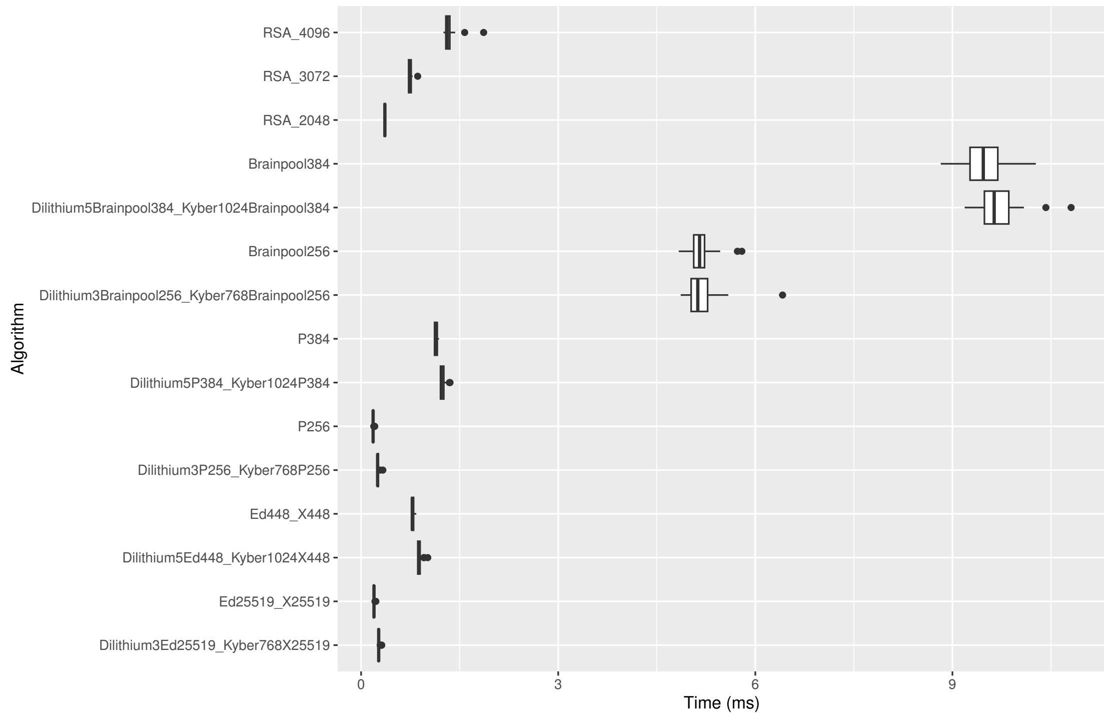


Figure 5.15: Time comparison of verification on ARM

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub



(a) x86 implementation



(b) ARM implementation

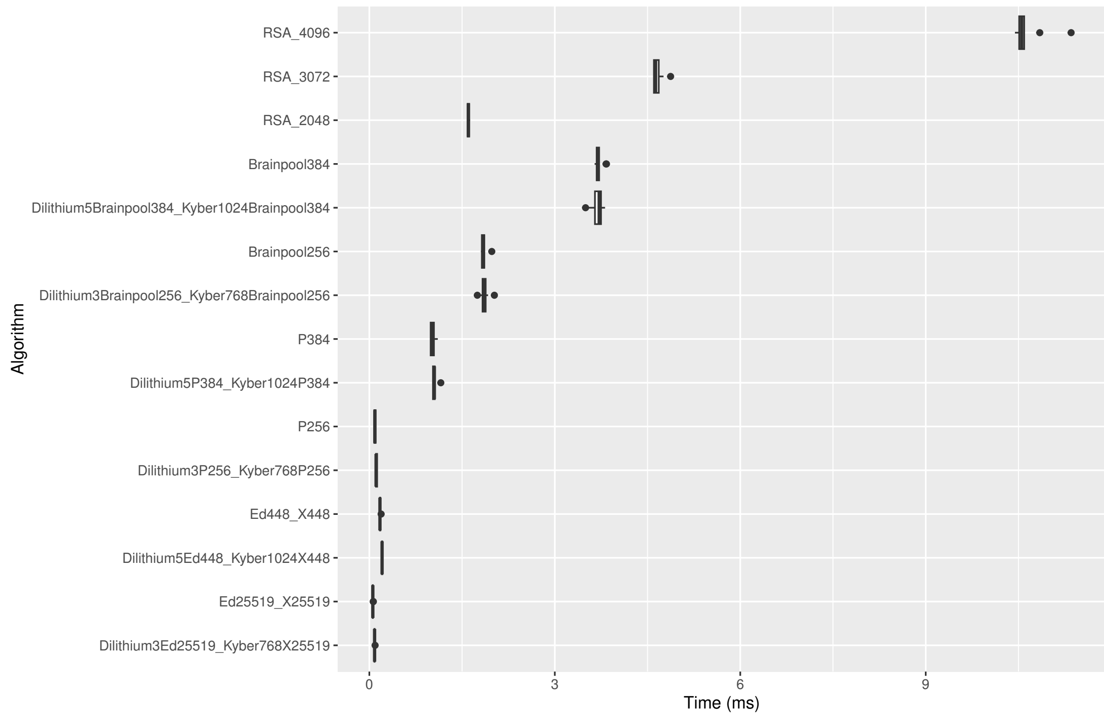
Figure 5.16: Time comparison of encryption

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Wien Bibliothek Your knowledge hub

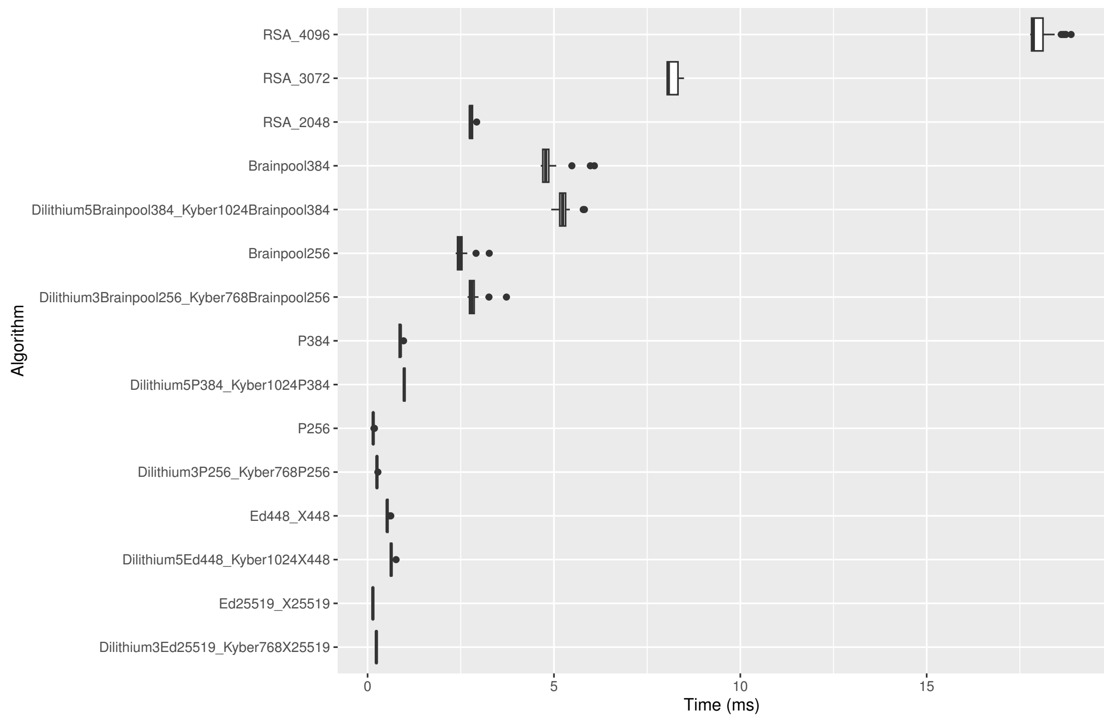
5.9 Decryption

Decryption shows a very similar picture to encryption, where overhead is barely visible in fig. 5.17. Also here, for X25519 decrypts in 0.09 ms in hybrid mode versus 0.06 ms as standalone on x86. All other elliptic curves are also in this case at least an order of magnitude slower than the lattice operations, making the hybrid and traditional measurements not significantly different.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub



(a) x86 implementation



(b) ARM implementation

Figure 5.17: Time comparison of decryption

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

CHAPTER 6

Conclusion

This project was developed with the intention to show a feasible and secure way to include PQ algorithms into the OpenPGP protocol, providing input to the standardization process for the working group and the community. The final aim is for this proposal to be integrated with feedback from other implementers and turn into a standards track RFC.

After selecting CRYSTALS-Kyber, CRYSTALS-Dilithium, and SPHINCS$^+$ from the NIST standardization process we have constructed 14 algorithm identifiers made of composite combinations: the CRYSTALS-based algorithms with the most common elliptic curves, and SPHINCS$^+$ standalone with two different underlying hashes.

To encrypt messages, we selected to use KEMs, in particular CRYSTALS-Kyber hybrid with X25519, X448, P-256, P-384, Brainpool P-256, or Brainpool P-384. These curves were selected because they are already supported by OpenPGP and ensure the widest range of compliance and compatibility with existing hardware. The secrets derived from Kyber and the curve are then combined using a KMAC-based KDF and key combiner that preserves the IND–CCA2 security of each KEM.

To sign messages, we selected SPHINCS$^+$ as standalone or CRYSTALS-Dilithium hybrid with Ed25519, Ed448, P-256, P-384, Brainpool P-256, or Brainpool P-384; the first two curves with EdDSA, while the latter four with ECDSA. This choice was dictated from the different use cases of the protocol: some users desire a high security margin for long-term keys, and are willing to sacrifice performance, while some others require fast operations. SPHINCS$^+$ is in fact a very conservative choice, that offers a solid security proof at the expense of slower operations. To ease the burden of the tradeoff made with SPHINCS$^+$ we decided to implement two different underlying hashes and parametrize it, allowing implementations and users to determine how critical is speed in their usage: OpenPGP ranges from software signing to real-time chat.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

We have picked fixed combinations over the freely composable variant to ensure a simpler and more robust construction, that requires less changes to the protocol and a simpler interface, greatly reducing complexity. This has almost removed the need for protocol changes, limiting them to hash binding and optional signalling mechanisms. All the PQ algorithms have been implemented by extending the existing algorithm identifiers and keeping the logic and artifacts within the algorithm-specific part.

A a proof-of-concept implementation was developed starting from `go-crypto`, a Golang library implementing OpenPGP, using existing libraries to provide cryptographic primitives. This has proven to be relatively straightforward given the fixed combination design, requiring only to implement the new algorithms and some minor changes to the protocol level implementation.

The performance of the implementation was tested to compare it with the existing algorithms on desktop devices and Android phones, compiling it via gomobile.

6.1 Results

While artifacts from PQ algorithms are significantly larger than their traditional counterparts, they do not seem to be problematic for most OpenPGP use cases. In particular, email can already handle the new keys, ciphertexts, and signatures seamlessly. Some considerations need to be done when scaling: while significantly larger artifacts do not affect the single message, they may be problematic for providers deploying this standard to million of messages. An edge case is the fast variant of SPHINCS$^+$, that presents signatures over 10KB in size. They will require re-thinking of the user experience, especially when signing small ciphertexts.

Regarding performance, the lattice-based algorithms add just a small overhead, keeping the hybrid schemes' perfomance still significantly above RSA. Since RSA is still widely used, we can assert that this should not imply any user experience drawback. In particular for encryption the overhead is minimal, and the efficiency of EC is preserved. On the other hand, SPHINCS$^+$ can present some user experience challenges when signing or generating new certificates. These will need to be handled from the application layer when switching to PQ algorithms: the implementation may require up to 3s for signing and up to 20s for key generation, therefore an indicator of progress may be necessary. All things considered, in particular for the recommended algorithms, the implementation has a similar performance profile to today's operations, allowing for a transparent migration from the user's perspective.

6.2 Directions for Future Research

6.2.1 Standardization

This project was developed in collaboration with the community, in order to pave the way for standardization into an IETF RFC. The following steps will be determined at

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

the upcoming meetings, where the Working Group (WG) will be rechartered and this draft will be proposed for adoption.

6.2.2 Expanding the Algorithm Selection

To provide a higher degree of security in the OpenPGP environment, a second PQ KEM could be standardized, based on a different underlying problem than Kyber. In particular, code theory based algorithms such as BIKE are considered very interesting, if any standardization body would consider them ready and provide a standard to follow. Since OpenPGP presents a serious legacy burden, and once support for an algorithm is added it is very difficult to deprecate it, implementation should strictly follow standardization: this prevents having to support non-standard implementations forever. The publication of another algorithm set can in any case follow in a different RFC.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

APPENDIX A

Integral RFC Text

In this appendix we display the integral text of the draft RFC published on March 25th 2023 on the IETF Internet-Draft archive. Version 01 specifies the normative text for the analysis carried out in this thesis. A draft RFC, being a living document, has since been updated to match implementer and security analysis feedback from the community.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Network Working Group
Internet-Draft
Intended status: Informational
Expires: 26 September 2023

S. Kousidis
BSI
F. Strenzke
MTG AG
A. Wussler
Proton AG
25 March 2023

Post-Quantum Cryptography in OpenPGP
draft-wussler-openpgp-pqc-01

Abstract

This document defines a post-quantum public-key algorithm extension for the OpenPGP protocol. Given the generally assumed threat of a cryptographically relevant quantum computer, this extension provides a basis for long-term secure OpenPGP signatures and ciphertexts. Specifically, it defines composite public-key encryption based on CRYSTALS-Kyber, composite public-key signatures based on CRYSTALS-Dilithium, both in combination with elliptic curve cryptography, and SPHINCS+ as a standalone public key signature scheme.

About This Document

This note is to be removed before publishing as an RFC.

Status information for this document may be found at https://datatracker.ietf.org/doc/draft-wussler-openpgp-pqc/.

Discussion of this document takes place on the WG Working Group mailing list (mailto:openpgp@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/openpgp/. Subscribe at https://www.ietf.org/mailman/listinfo/openpgp/.

Source for this draft and an issue tracker can be found at https://github.com/openpgp-pqc/draft-openpgp-pqc.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 26 September 2023.

Copyright Notice

Copyright (c) 2023 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction 4
  1.1. Conventions used in this Document 4
    1.1.1. Terminology for Multi-Algorithm Schemes 5
  1.2. Post-Quantum Cryptography 5
    1.2.1. CRYSTALS-Kyber 5
    1.2.2. CRYSTALS-Dilithium 6
    1.2.3. SPHINCS+ 6
  1.3. Elliptic Curve Cryptography 6
    1.3.1. Curve25519 and Curve448 6
    1.3.2. Generic Prime Curves 6
  1.4. Standalone and Multi-Algorithm Schemes 7
    1.4.1. Standalone and Composite Multi-Algorithm Schemes . . 7
    1.4.2. Non-Composite Algorithm Combinations 7
2. Preliminaries 8
  2.1. Elliptic curves 8
    2.1.1. SEC1 EC Point Wire Format 8
    2.1.2. Measures to Ensure Secure Implementations 8
3. Supported Public Key Algorithms 9
  3.1. Algorithm Specifications 9
  3.2. Parameter Specification 10
    3.2.1. SPHINCS+-simple-SHA2 10
    3.2.2. SPHINCS+-simple-SHAKE 11
4. Algorithm Combinations 12
  4.1. Composite KEMs 12
  4.2. Parallel Public-Key Encryption 12

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

```
Kousidis, et al.       Expires 26 September 2023              [Page 2]

Internet-Draft              PQC in OpenPGP                  March 2023
```


```
     4.3.  Composite Signatures  . . . . . . . . . . . . . . . . . .  12
     4.4.  Multiple Signatures . . . . . . . . . . . . . . . . . . .  12
   5.  Composite KEM schemes . . . . . . . . . . . . . . . . . . . .  13
     5.1.  Building Blocks . . . . . . . . . . . . . . . . . . . . .  13
       5.1.1.  ECC-Based KEMs  . . . . . . . . . . . . . . . . . . .  13
       5.1.2.  Kyber-KEM . . . . . . . . . . . . . . . . . . . . . .  17
     5.2.  Composite Encryption Schemes with Kyber . . . . . . . . .  18
       5.2.1.  Fixed information . . . . . . . . . . . . . . . . . .  19
       5.2.2.  Key combiner  . . . . . . . . . . . . . . . . . . . .  20
       5.2.3.  Key generation procedure  . . . . . . . . . . . . . .  21
       5.2.4.  Encryption procedure  . . . . . . . . . . . . . . . .  21
       5.2.5.  Decryption procedure  . . . . . . . . . . . . . . . .  22
     5.3.  Packet specifications . . . . . . . . . . . . . . . . . .  23
       5.3.1.  Public-Key Encrypted Session Key Packets (Tag 1)  . .  23
       5.3.2.  Key Material Packets  . . . . . . . . . . . . . . . .  23
   6.  Composite Signature Schemes . . . . . . . . . . . . . . . . .  24
     6.1.  Building blocks . . . . . . . . . . . . . . . . . . . . .  24
       6.1.1.  EdDSA-Based signatures  . . . . . . . . . . . . . . .  24
       6.1.2.  ECDSA-Based signatures  . . . . . . . . . . . . . . .  24
       6.1.3.  Dilithium signatures  . . . . . . . . . . . . . . . .  25
     6.2.  Composite Signature Schemes with Dilithium  . . . . . . .  26
       6.2.1.  Binding hashes  . . . . . . . . . . . . . . . . . . .  26
       6.2.2.  Key generation procedure  . . . . . . . . . . . . . .  26
       6.2.3.  Signature Generation  . . . . . . . . . . . . . . . .  26
       6.2.4.  Signature Verification  . . . . . . . . . . . . . . .  27
     6.3.  Packet Specifications . . . . . . . . . . . . . . . . . .  27
       6.3.1.  Signature Packet (Tag 2)  . . . . . . . . . . . . . .  27
       6.3.2.  Key Material Packets  . . . . . . . . . . . . . . . .  28
   7.  SPHINCS+  . . . . . . . . . . . . . . . . . . . . . . . . . .  29
     7.1.  The SPHINCS+ Algorithms . . . . . . . . . . . . . . . . .  29
       7.1.1.  Binding hashes  . . . . . . . . . . . . . . . . . . .  30
       7.1.2.  Key generation  . . . . . . . . . . . . . . . . . . .  30
       7.1.3.  Signature Generation  . . . . . . . . . . . . . . . .  30
       7.1.4.  Signature Verification  . . . . . . . . . . . . . . .  31
     7.2.  Packet specifications . . . . . . . . . . . . . . . . . .  31
       7.2.1.  Signature Packet (Tag 2)  . . . . . . . . . . . . . .  31
       7.2.2.  Key Material Packets  . . . . . . . . . . . . . . . .  31
   8.  Migration Considerations  . . . . . . . . . . . . . . . . . .  32
     8.1.  Key preference  . . . . . . . . . . . . . . . . . . . . .  32
     8.2.  Key generation strategies . . . . . . . . . . . . . . . .  32
   9.  Security Considerations . . . . . . . . . . . . . . . . . . .  33
     9.1.  Hashing in ECC-KEM  . . . . . . . . . . . . . . . . . . .  33
     9.2.  Key combiner  . . . . . . . . . . . . . . . . . . . . . .  33
     9.3.  Domain separation and binding . . . . . . . . . . . . . .  34
     9.4.  SPHINCS+  . . . . . . . . . . . . . . . . . . . . . . . .  34
     9.5.  Binding hashes in signatures with signature algorithms .  35
   10. Additional considerations . . . . . . . . . . . . . . . . . .  35
     10.1.  Performance Considerations for SPHINCS+  . . . . . . . .  35
```


Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

11. IANA Considerations 35
12. Contributors 36
13. References 36
13.1. Normative References 36
13.2. Informative References 36
Acknowledgments 38
Authors' Addresses 39

1. Introduction

The OpenPGP protocol supports various traditional public-key algorithms based on the factoring or discrete logarithm problem. As the security of algorithms based on these mathematical problems is endangered by the advent of quantum computers, there is a need to extend OpenPGP by algorithms that remain secure in the presence of quantum computers.

Such cryptographic algorithms are referred to as post-quantum cryptography. The algorithms defined in this extension were chosen for standardization by the National Institute of Standards and Technology (NIST) in mid 2022 [NISTIR-8413] as the result of the NIST Post-Quantum Cryptography Standardization process initiated in 2016 [NIST-PQC]. Namely, these are CRYSTALS-Kyber as a Key Encapsulation Mechanism (KEM), a KEM being a modern building block for public-key encryption, and CRYSTALS-Dilithium as well as SPHINCS+ as signature schemes.

For the two CRYSTALS-* schemes, this document follows the conservative strategy to deploy post-quantum in combination with traditional schemes such that the security is retained even if all schemes but one in the combination are broken. In contrast, the hashed-based signature scheme SPHINCS+ is considered to be sufficiently well understood with respect to its security assumptions in order to be used standalone. To this end, this document specifies the following new set: SPHINCS+ standalone and CRYSTALS-* as composite with ECC-based KEM and digital signature schemes. Here, the term "composite" indicates that any data structure or algorithm pertaining to the combination of the two components appears as single data structure or algorithm from the protocol perspective.

The document specifies the conventions for interoperability between compliant OpenPGP implementations that make use of this extension and the newly defined algorithms or algorithm combinations.

1.1. Conventions used in this Document

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

1.1.1. Terminology for Multi-Algorithm Schemes

The terminology in this document is oriented towards the definitions in [draft-driscoll-pqt-hybrid-terminology]. Specifically, the terms "multi-algorithm", "composite" and "non-composite" are used in correspondence with the definitions therein. The abbreviation "PQ" is used for post-quantum schemes. To denote the combination of post-quantum and traditional schemes, the abbreviation "PQ/T" is used. The short form "PQ(/T)" stands for PQ or PQ/T.

1.2. Post-Quantum Cryptography

This section describes the individual post-quantum cryptographic schemes. All schemes listed here are believed to provide security in the presence of a cryptographically relevant quantum computer. However, the mathematical problems on which the two CRYSTALS-* schemes and SPHINCS+ are based, are fundamentally different, and accordingly the level of trust commonly placed in them as well as their performance characteristics vary.

[Note to the reader: This specification refers to the latest NIST submission papers of each scheme as if it were a specification. This is a temporary solution that is owed to the fact that currently no other specification is available. The goal is to provide a sufficiently precise specification of the algorithms already at the draft stage of this specification, so that it is possible for implementers to create interoperable implementations. As soon as standards by NIST or the IETF for the PQC schemes employed in this specification are available, these will replace the references to the NIST submission papers. Furthermore, we want to point out that, depending on possible changes to the schemes standardized by NIST, this specification may be updated substantially as soon as corresponding information becomes available.]

1.2.1. CRYSTALS-Kyber

CRYSTALS-Kyber [KYBER-Subm] is based on the hardness of solving the learning-with-errors problem in module lattices (MLWE). The scheme is believed to provide security against cryptanalytic attacks by classical as well as quantum computers. This specification defines CRYSTALS-Kyber only in composite combination with ECC-based encryption schemes in order to provide a pre-quantum security fallback.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

1.2.2. CRYSTALS-Dilithium

CRYSTALS-Dilithium, defined in [DILITHIUM-Subm], is a signature scheme that, like CRYSTALS-Kyber, is based on the hardness of solving lattice problems in module lattices. Accordingly, this specification only defines CRYSTALS-Dilithium in composite combination with ECC-based signature schemes.

1.2.3. SPHINCS+

SPHINCS+ [SPHINCS-Subm] is a stateless hash-based signature scheme. Its security relies on the hardness of finding preimages for cryptographic hash functions. This feature is generally considered to be a high security guarantee. Therefore, this specification defines SPHINCS+ as a standalone signature scheme.

In deployments the performance characteristics of SPHINCS+ should be taken into account. We refer to Section 10.1 for a discussion of the performance characteristics of this scheme.

1.3. Elliptic Curve Cryptography

The ECC-based encryption is defined here as a KEM. This is in contrast to [I-D.ietf-openpgp-crypto-refresh] where the ECC-based encryption is defined as a public-key encryption scheme.

All elliptic curves for the use in the composite combinations are taken from [I-D.ietf-openpgp-crypto-refresh]. However, as explained in the following, in the case of Curve25519 encoding changes are applied to the new composite schemes.

1.3.1. Curve25519 and Curve448

Curve25519 and Curve448 are defined in [RFC7748] for use in a Diffie-Hellman key agreement scheme and defined in [RFC8032] for use in a digital signature scheme. For Curve25519 this specification adapts the encoding of objects as defined in [RFC7748] in contrast to [I-D.ietf-openpgp-crypto-refresh].

1.3.2. Generic Prime Curves

For interoperability this extension offers CRYSTALS-* in composite combinations with the NIST curves P-256, P-384 defined in [SP800-186] and the Brainpool curves brainpoolP256r1, brainpoolP384r1 defined in [RFC5639].

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

1.4. Standalone and Multi-Algorithm Schemes

This section provides a categorization of the new algorithms and their combinations.

1.4.1. Standalone and Composite Multi-Algorithm Schemes

This specification introduces new cryptographic schemes, which can be categorized as follows:

* PQ/T multi-algorithm public-key encryption, namely a composite combination of CRYSTALS-Kyber with an ECC-based KEM,

* PQ/T multi-algorithm digital signature, namely composite combinations of CRYSTALS-Dilithium with ECC-based signature schemes,

* PQ digital signature, namely SPHINCS+ as a standalone cryptographic algorithm.

For each of the composite schemes, this specifications mandates that the recipient has to successfully perform the cryptographic algorithms for each of the component schemes used in a cryptrographic message, in order for the message to be deciphered and considered as valid. This means that all component signatures must be verified successfully in order to achieve a successful verification of the composite signature. In the case of the composite public-key decryption, each of the component KEM decapsulation operations must succeed.

1.4.2. Non-Composite Algorithm Combinations

As the OpenPGP protocol [I-D.ietf-openpgp-crypto-refresh] allows for multiple signatures to be applied to a single message, it is also possible to realize non-composite combinations of signatures. Furthermore, multiple OpenPGP signatures may be combined on the application layer. These latter two cases realize non-composite combinations of signatures. Section 4.4 specifies how implementations should handle the verification of such combinations of signatures.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Furthermore, the OpenPGP protocol also allows for parallel encryption to different keys held by the same recipient. Accordingly, if the sender makes use of this feature and sends an encrypted message with multiple PKESK packages for different encryption keys held by the same recipient, a non-composite multi-algorithm public-key encryption is realized where the recipient has to decrypt only one of the PKESK packages in order to decrypt the message. See Section 4.2 for restrictions on parallel encryption mandated by this specification.

2. Preliminaries

This section provides some preliminaries for the definitions in the subsequent sections.

2.1. Elliptic curves

2.1.1. SEC1 EC Point Wire Format

Elliptic curve points of the generic prime curves are encoded using the SEC1 (uncompressed) format as the following octet string:

B = 04 || X || Y

where X and Y are coordinates of the elliptic curve point P = (X, Y), and each coordinate is encoded in the big-endian format and zero-padded to the adjusted underlying field size. The adjusted underlying field size is the underlying field size rounded up to the nearest 8-bit boundary, as noted in the "Field size" column in Table 6, Table 7, or Table 11. This encoding is compatible with the definition given in [SEC1].

2.1.2. Measures to Ensure Secure Implementations

The following paragraphs describe measures that ensure secure implementations according to existing best practices and standards defining the operations of Elliptic Curve Cryptography.

Even though the zero point, also called the point at infinity, may occur as a result of arithmetic operations on points of an elliptic curve, it MUST NOT appear in any ECC data structure defined in this document.

Furthermore, when performing the explicitly listed operations in Section 5.1.1.1, Section 5.1.1.2 or Section 5.1.1.3 it is REQUIRED to follow the specification and security advisory mandated from the relative elliptic curve specification.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

3. Supported Public Key Algorithms

This section specifies the composite Kyber + ECC and Dilithium + ECC schemes as well as the standalone SPHINCS+ signature scheme. The composite schemes are fully specified via their algorithm ID. The SPHINCS+ signature schemes are fully specified by their algorithm ID and an additional parameter ID.

3.1. Algorithm Specifications

For encryption, the following composite KEM schemes are specified:

ID	Algorithm	Requirement	Definition
29	Kyber768 + X25519	MUST	Section 5.2
30	Kyber1024 + X448	SHOULD	Section 5.2
31	Kyber768 + ECDH-NIST-P-256	MAY	Section 5.2
32	Kyber1024 + ECDH-NIST-P-384	MAY	Section 5.2
33	Kyber768 + ECDH-brainpoolP256r1	MAY	Section 5.2
34	Kyber1024 + ECDH-brainpoolP384r1	MAY	Section 5.2

Table 1: KEM algorithm specifications

For signatures, the following (composite) signature schemes are specified:

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

ID	Algorithm	Requirement	Definition
35	Dilithium3 + Ed25519	MUST	Section 6.2
36	Dilithium5 + Ed448	SHOULD	Section 6.2
37	Dilithium3 + ECDSA-NIST-P-256	MAY	Section 6.2
38	Dilithium5 + ECDSA-NIST-P-384	MAY	Section 6.2
39	Dilithium3 + ECDSA-brainpoolP256r1	MAY	Section 6.2
40	Dilithium5 + ECDSA-brainpoolP384r1	MAY	Section 6.2
41	SPHINCS+-simple-SHA2	SHOULD	Section 1.2.3
42	SPHINCS+-simple-SHAKE	MAY	Section 1.2.3

Table 2: Signature algorithm specifications

3.2. Parameter Specification

3.2.1. SPHINCS+-simple-SHA2

For the SPHINCS+-simple-SHA2 signature algorithm from Table 2, the following parameters are specified:

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Parameter ID	Parameter
1	SPHINCS+-simple-SHA2-128s
2	SPHINCS+-simple-SHA2-128f
3	SPHINCS+-simple-SHA2-192s
4	SPHINCS+-simple-SHA2-192f
5	SPHINCS+-simple-SHA2-256s
6	SPHINCS+-simple-SHA2-256f

Table 3: SPHINCS+-simple-SHA2 security parameters

All security parameters inherit the requirement of SPHINCS+-simple-SHA2 from Table 2. That is, implementations SHOULD implement the parameters specified in Table 3. The values 0x00 and 0xFF are reserved for future extensions.

3.2.2. SPHINCS+-simple-SHAKE

For the SPHINCS+-simple-SHAKE signature algorithm from Table 2, the following parameters are specified:

Parameter ID	Parameter
1	SPHINCS+-simple-SHAKE-128s
2	SPHINCS+-simple-SHAKE-128f
3	SPHINCS+-simple-SHAKE-192s
4	SPHINCS+-simple-SHAKE-192f
5	SPHINCS+-simple-SHAKE-256s
6	SPHINCS+-simple-SHAKE-256f

Table 4: SPHINCS+-simple-SHAKE security parameters

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

All security parameters inherit the requirement of SPHINCS+-simple-SHAKE from Table 2. That is, implementations MAY implement the parameters specified in Table 4. The values 0x00 and 0xFF are reserved for future extensions.

4. Algorithm Combinations

4.1. Composite KEMs

Kyber + ECC public-key encryption is meant to involve both the Kyber KEM and an ECC-based KEM in an a priori non-separable manner. This is achieved via KEM combination, i.e. both key encapsulations/decapsulations are performed in parallel, and the resulting key shares are fed into a key combiner to produce a single shared secret for message encryption.

4.2. Parallel Public-Key Encryption

As explained in Section 1.4.2, the OpenPGP protocol inherently supports parallel encryption to different keys of the same recipient. Implementations MUST NOT encrypt a message to a purely traditional public-key encryption key of a recipient if it is encrypted to a PQ/T key of the same recipient.

4.3. Composite Signatures

Dilithium + ECC signatures are meant to contain both the Dilithium and the ECC signature data, and an implementation MUST validate both algorithms to state that a signature is valid.

4.4. Multiple Signatures

The OpenPGP message format allows multiple signatures of a message, i.e. the attachment of multiple signature packets.

An implementation MAY sign a message with a traditional key and a PQ(/T) key from the same sender. This ensures backwards compatibility due to [I-D.ietf-openpgp-crypto-refresh] Section 5.2.5, since a legacy implementation without PQ(/T) support can fall back on the traditional signature.

Newer implementations with PQ(/T) support MAY ignore the traditional signature(s) during validation.

Implementations SHOULD consider the message correctly signed if at least one of the non-ignored signatures validates successfully.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

[Note to the reader: The last requirement, that one valid signature is sufficient to identify a message as correctly signed, is an interpretation of [I-D.ietf-openpgp-crypto-refresh] Section 5.2.5.]

5. Composite KEM schemes

5.1. Building Blocks

5.1.1. ECC-Based KEMs

In this section we define the encryption, decryption, and data formats for the ECDH component of the composite algorithms.

Table 5, Table 6, and Table 7 describe the ECC-KEM parameters and artifact lengths. The artefacts in Table 5 follow the encodings described in [RFC7748].

	X25519	X448
Algorithm ID reference	29	30
Field size	32 octets	56 octets
ECC-KEM	x25519Kem (Section 5.1.1.1)	x448Kem (Section 5.1.1.2)
ECDH public key	32 octets [RFC7748]	56 octets [RFC7748]
ECDH secret key	32 octets [RFC7748]	56 octets [RFC7748]
ECDH ephemeral	32 octets [RFC7748]	56 octets [RFC7748]
ECDH share	32 octets [RFC7748]	56 octets [RFC7748]
Key share	32 octets	64 octets
Hash	SHA3-256	SHA3-512

Table 5: Montgomery curves parameters and artifact lengths

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

	NIST P-256	NIST P-384
Algorithm ID reference	31	32
Field size	32 octets	48 octets
ECC-KEM	ecdhKem (Section 5.1.1.3)	ecdhKem (Section 5.1.1.3)
ECDH public key	65 octets of SEC1-encoded public point	97 octets of SEC1-encoded public point
ECDH secret key	32 octets big-endian encoded secret scalar	48 octets big-endian encoded secret scalar
ECDH ephemeral	65 octets of SEC1-encoded ephemeral point	97 octets of SEC1-encoded ephemeral point
ECDH share	65 octets of SEC1-encoded shared point	97 octets of SEC1-encoded shared point
Key share	32 octets	64 octets
Hash	SHA3-256	SHA3-512

Table 6: NIST curves parameters and artifact lengths

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

	brainpoolP256r1	brainpoolP384r1
Algorithm ID reference	33	34
Field size	32 octets	48 octets
ECC-KEM	ecdhKem (Section 5.1.1.3)	ecdhKem (Section 5.1.1.3)
ECDH public key	65 octets of SEC1-encoded public point	97 octets of SEC1-encoded public point
ECDH secret key	32 octets big-endian encoded secret scalar	48 octets big-endian encoded secret scalar
ECDH ephemeral	65 octets of SEC1-encoded ephemeral point	97 octets of SEC1-encoded ephemeral point
ECDH share	65 octets of SEC1-encoded shared point	97 octets of SEC1-encoded shared point
Key share	32 octets	64 octets
Hash	SHA3-256	SHA3-512

Table 7: Brainpool curves parameters and artifact lengths

The SEC1 format for point encoding is defined in Section 2.1.1.

The various procedures to perform the operations of an ECC-based KEM are defined in the following subsections. Specifically, each of these subsections defines the instances of the following operations:

```
(eccCipherText, eccKeyShare) <- eccKem.encap(eccPublicKey)
```

and

```
(eccKeyShare) <- eccKem.decap(eccPrivateKey, eccCipherText)
```

The placeholder eccKem has to be replaced with the specific ECC-KEM from the row "ECC-KEM" of Table 5, Table 6, and Table 7.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

5.1.1.1. X25519-KEM

The encapsulation and decapsulation operations of x25519kem are described using the function X25519() and encodings defined in [RFC7748]. The eccPrivateKey is denoted as r, the eccPublicKey as R, they are subject to the equation R = X25519(r, U(P)). Here, U(P) denotes the u-coordinate of the base point of Curve25519.

The operation x25519Kem.encap() is defined as follows:

1. Generate an ephemeral key pair {v, V} via V = X25519(v,U(P))

2. Compute the shared coordinate X = X25519(v, R) where R is the public key eccPublicKey

3. Set the output eccCipherText to V

4. Set the output eccKeyShare to SHA3-256(X || eccCipherText)

The operation x25519Kem.decap() is defined as follows:

1. Compute the shared coordinate X = X25519(r, V), where r is the eccPrivateKey and V is the eccCipherText

2. Set the output eccKeyShare to SHA3-256(X || eccCipherText)

5.1.1.2. X448-KEM

The encapsulation and decapsulation operations of x448kem are described using the function X448() and encodings defined in [RFC7748]. The eccPrivateKey is denoted as r, the eccPublicKey as R, they are subject to the equation R = X25519(r, U(P)). Here, U(P) denotes the u-coordinate of the base point of Curve448.

The operation x448.encap() is defined as follows:

1. Generate an ephemeral key pair {v, V} via V = X448(v,U(P))

2. Compute the shared coordinate X = X448(v, R) where R is the public key eccPublicKey

3. Set the output eccCipherText to V

4. Set the output eccKeyShare to SHA3-512(X || eccCipherText)

The operation x448Kem.decap() is defined as follows:

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

```
   1.  Compute the shared coordinate X = X448(r, V), where r is the
       eccPrivateKey and V is the eccCipherText

   2.  Set the output eccKeyShare to SHA3-512(X || eccCipherText)
```

5.1.1.3. ECDH-KEM

```
   The operation ecdhKem.encap() is defined as follows:

   1.  Generate an ephemeral key pair {v, V=vG} as defined in
       [SP800-186] or [RFC5639]

   2.  Compute the shared point S = vR, where R is the component public
       key eccPublicKey, according to [SP800-186] or [RFC5639]

   3.  Extract the X coordinate from the SEC1 encoded point S = 04 ||
       X || Y as defined in section Section 2.1.1

   4.  Set the output eccCipherText to the SEC1 encoding of V

   5.  Set the output eccKeyShare to Hash(X || eccCipherText), with Hash
       chosen according to Table 6 or Table 7

   The operation ecdhKem.decap() is defined as follows:

   1.  Compute the shared Point S as rV, where r is the eccPrivateKey
       and V is the eccCipherText, according to [SP800-186] or [RFC5639]

   2.  Extract the X coordinate from the SEC1 encoded point S = 04 ||
       X || Y as defined in section Section 2.1.1

   3.  Set the output eccKeyShare to Hash(X || eccCipherText), with Hash
       chosen according to Table 6 or Table 7
```

5.1.2. Kyber-KEM

```
   Kyber-KEM features the following operations:

   (kyberCipherText, kyberKeyShare) <- kyberKem.encap(kyberPublicKey)

   and

   (kyberKeyShare) <- kyberKem.decap(kyberCipherText, kyberPrivateKey)

   The above are the operations Kyber.CCAKEM.Enc() and
   Kyber.CCAKEM.Dec() defined in [KYBER-Subm].
```


Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Kyber-KEM has the parameterization with the corresponding artifact lengths in octets as given in Table 8. All artifacts are encoded as defined in [KYBER-Subm].

Algorithm ID reference	Kyber-KEM	Public key	Secret key	Ciphertext	Key share
29, 31, 33	kyberKem768	1184	2400	1088	32
30, 32, 34	kyberKem1024	1568	3186	1568	32

Table 8: Kyber-KEM parameters artifact lengths in octets

The placeholder kyberKem has to be replaced with the specific Kyber-KEM from the column "Kyber-KEM" of Table 8.

The procedure to perform kyberKem.encap() is as follows:

1. Extract the component public key kyberPublicKey that is part of the recipient's composite public key

2. Invoke (kyberCipherText, keyShare) <- kyberKem.encap(kyberPublicKey)

3. Set kyberCipherText as the Kyber ciphertext

4. Set keyShare as the Kyber symmetric key share

The procedure to perform kyberKem.decap() is as follows:

1. Invoke keyShare <- kyberKem.decap(kyberCipherText, kyberPrivateKey)

2. Set keyShare as the Kyber symmetric key

5.2. Composite Encryption Schemes with Kyber

Table 1 specifies the following Kyber + ECC composite public-key encryption schemes:

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Algorithm ID reference	Kyber-KEM	ECC-KEM	ECDH-KEM curve
29	kyberKem768	x25519Kem	X25519
30	kyberKem1024	x448Kem	X448
31	kyberKem768	ecdhKem	NIST P-256
32	kyberKem1024	ecdhKem	NIST P-384
33	kyberKem768	ecdhKem	brainpoolP256r1
34	kyberKem1024	ecdhKem	brainpoolP384r1

Table 9: Kyber-ECC-composite Schemes

The Kyber + ECC composite public-key encryption schemes are built according to the following principal design:

* The Kyber-KEM encapsulation algorithm is invoked to create a Kyber ciphertext together with a Kyber symmetric key share.

* The encapsulation algorithm of an ECC-based KEM, namely one out of X25519-KEM, X448-KEM, or ECDH-KEM is invoked to create an ECC ciphertext together with an ECC symmetric key share.

* A Key-Encryption-Key (KEK) is computed as the output of a key combiner that receives as input both of the above created symmetric key shares and the protocol binding information.

* The session key for content encryption is then wrapped as described in [RFC3394] using AES-256 as algorithm and the KEK as key.

* The v6 PKESK package’s algorithm specific parts are made up of the Kyber ciphertext, the ECC ciphertext, and the wrapped session key

5.2.1. Fixed information

For the composite KEM schemes defined in Table 1 the following procedure, justified in Section 9.3, MUST be used to derive a string to use as binding between the KEK and the communication parties.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek, Your knowledge hub, WIEN

```
//   Input:
//   algID     - the algorithm ID encoded as octet
//   publicKey - the recipient's encryption sub-key packet
//               serialized as octet string

fixedInfo = algID || SHA3-256(publicKey)
```

SHA3-256 MUST be used to hash the publicKey of the recipient.

5.2.2. Key combiner

For the composite KEM schemes defined in Table 1 the following procedure MUST be used to compute the KEK that wraps a session key. The construction is a one-step key derivation function compliant to [SP800-56C] Section 4, based on KMAC256 [SP800-185]. It is given by the following algorithm.

```
//   multiKeyCombine(eccKeyShare, eccCipherText,
//                   kyberKeyShare, kyberCipherText,
//                   fixedInfo, oBits)
//
//   Input:
//   eccKeyShare     - the ECC key share encoded as an octet string
//   eccCipherText   - the ECC ciphertext encoded as an octet string
//   kyberKeyShare   - the Kyber key share encoded as an octet string
//   kyberCipherText - the Kyber ciphertext encoded as an octet string
//   fixedInfo       - the fixed information octet string
//   oBits           - the size of the output keying material in bits
//
//   Constants:
//   domSeparation       - the UTF-8 encoding of the string
//                         "OpenPGPCompositeKeyDerivationFunction"
//   counter             - the fixed 4 byte value 0x00000001
//   customizationString - the UTF-8 encoding of the string "KDF"

eccKemData = eccKeyShare || eccCipherText
kyberKemData = kyberKeyShare || kyberCipherText
encData = counter || eccKemData || kyberKemData || fixedInfo

MB = KMAC256(domSeparation, encData, oBits, customizationString)
```

Note that the values eccKeyShare defined in Section 5.1.1 and kyberKeyShare defined in Section 5.1.2 already use the relative ciphertext in the derivation. The ciphertext is by design included again in the key combiner to provide a robust security proof.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

The value of domSeparation is the UTF-8 encoding of the string "OpenPGPCompositeKeyDerivationFunction" and MUST be the following octet sequence:

```
domSeparation := 4F 70 65 6E 50 47 50 43 6F 6D 70 6F 73 69 74 65
                 4B 65 79 44 65 72 69 76 61 74 69 6F 6E 46 75 6E
                 63 74 69 6F 6E
```

The value of counter MUST be set to the following octet sequence:

```
counter :=  00 00 00 01
```

The value of fixedInfo MUST be set according to Section 5.2.1.

The value of customizationString is the UTF-8 encoding of the string "KDF" and MUST be set to the following octet sequence:

```
customizationString := 4B 44 46
```

5.2.3. Key generation procedure

The implementation MUST independently generate the Kyber and the ECC component keys. Kyber key generation follows the specification [KYBER-Subm] and the artifacts are encoded as fixed-length octet strings. For ECC this is done following the relative specification in [RFC7748], [SP800-186], or [RFC5639], and encoding the outputs as fixed-length octet strings in the format specified in table Table 5, Table 6, or Table 7.

5.2.4. Encryption procedure

The procedure to perform public-key encryption with a Kyber + ECC composite scheme is as follows:

1. Take the recipient’s authenticated public-key packet pkComposite and sessionKey as input

2. Parse the algorithm ID from pkComposite

3. Extract the eccPublicKey and kyberPublicKey component from the algorithm specific data encoded in pkComposite with the format specified in Section 5.3.2.

4. Instantiate the ECC-KEM eccKem.encap() and the Kyber-KEM kyberKem.encap() depending on the algorithm ID according to Table 9

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

5. Compute (eccCipherText, eccKeyShare) := eccKem.encap(eccPublicKey)

6. Compute (kyberCipherText, kyberKeyShare) := kyberKem.encap(kyberPublicKey)

7. Compute fixedInfo as specified in Section 5.2.1

8. Compute KEK := multiKeyCombine(eccKeyShare, eccCipherText, kyberKeyShare, kyberCipherText, fixedInfo, oBits=256) as defined in Section 5.2.2

9. Compute C := AESKeyWrap(KEK, sessionKey) with AES-256 as per [RFC3394] that includes a 64 bit integrity check

10. Output eccCipherText || kyberCipherText || len(C) || C as specified in Section 5.3.1

5.2.5. Decryption procedure

The procedure to perform public-key decryption with a Kyber + ECC composite scheme is as follows:

1. Take the matching PKESK and own secret key packet as input

2. From the PKESK extract the algorithm ID and the encryptedKey

3. Check that the own and the extracted algorithm ID match

4. Parse the eccSecretKey and kyberSecretKey from the algorithm specific data of the own secret key encoded in the format specified in Section 5.3.2

5. Instantiate the ECC-KEM eccKem.decap() and the Kyber-KEM kyberKem.decap() depending on the algorithm ID according to Table 9

6. Parse eccCipherText, kyberCipherText, and C from encryptedKey encoded as eccCipherText || kyberCipherText || len(C) || C as specified in Section 5.3.1

7. Compute (eccKeyShare) := eccKem.decap(eccCipherText, eccPrivateKey)

8. Compute (kyberKeyShare) := kyberKem.decap(kyberCipherText, kyberPrivateKey)

9. Compute fixedInfo as specified in Section 5.2.1

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

10. Compute KEK := multiKeyCombine(eccKeyShare, eccCipherText, kyberKeyShare, kyberCipherText, fixedInfo, oBits=256) as defined in Section 5.2.2

11. Compute sessionKey := AESKeyUnwrap(KEK, C) with AES-256 as per [RFC3394], aborting if the 64 bit integrity check fails

12. Output sessionKey

5.3. Packet specifications

5.3.1. Public-Key Encrypted Session Key Packets (Tag 1)

The composite Kyber algorithms MUST be used only with v6 PKESK, as defined in [I-D.ietf-openpgp-crypto-refresh] Section 5.1.2.

The algorithm-specific v6 PKESK parameters consists of:

* A fixed-length octet string representing an ECC ephemeral public key in the format associated with the curve as specified in Section 5.1.1.

* A fixed-length octet string of the Kyber ciphertext, whose length depends on the algorithm ID as specified in Table 8.

* A variable-length field containing the symmetric key:

  - A one-octet size of the following field;

  - Octet string of the wrapped symmetric key as described in Section 5.2.4.

5.3.2. Key Material Packets

The algorithm-specific public key is this series of values:

* A fixed-length octet string representing an EC point public key, in the point format associated with the curve specified in Section 5.1.1.

* A fixed-length octet string containing the Kyber public key, whose length depends on the algorithm ID as specified in Table 8.

The algorithm-specific secret key is these two values:

* A fixed-length octet string of the encoded secret scalar, whose encoding and length depend on the algorithm ID as specified in Section 5.1.1.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

* A fixed-length octet string containing the Kyber secret key, whose length depends on the algorithm ID as specified in Table 8.

6. Composite Signature Schemes

6.1. Building blocks

6.1.1. EdDSA-Based signatures

To sign and verify with EdDSA the following operations are defined:

```
(eddsaSignature) <- eddsa.sign(eddsaPrivateKey, dataDigest)
```

and

```
(verified) <- eddsa.verify(eddsaPublicKey, eddsaSignature, dataDigest)
```

The public and private keys, as well as the signature MUST be encoded according to [RFC8032] as fixed-length octet strings. The following table describes the EdDSA parameters and artifact lengths:

Algorithm ID reference	Curve	Field size	Public key	Secret key	Signature
35	Ed25519	32	32	32	64
36	Ed448	57	57	57	114

Table 10: EdDSA parameters and artifact lengths in octets

6.1.2. ECDSA-Based signatures

To sign and verify with ECDSA the following operations are defined:

```
(ecdsaSignatureR, ecdsaSignatureS) <- ecdsa.sign(ecdsaPrivateKey,
                                                 dataDigest)
```

and

```
(verified) <- ecdsa.verify(ecdsaPublicKey, ecdsaSignatureR,
                           ecdsaSignatureS, dataDigest)
```

The public keys MUST be encoded in SEC1 format as defined in section Section 2.1.1. The secret key, as well as both values R and S of the signature MUST each be encoded as a big-endian integer in a fixed-length octet string of the specified size.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

The following table describes the ECDSA parameters and artifact lengths:

Algorithm ID reference	Curve	Field size	Public key	Secret key	Signature value R	Signature value S
37	NIST P-256	32	65	32	32	32
38	NIST P-384	48	97	48	48	48
39	brainpoolP256r1	32	65	32	32	32
40	brainpoolP384r1	48	97	48	48	48

Table 11: ECDSA parameters and artifact lengths in octets

6.1.3. Dilithium signatures

The procedure for Dilithium signature generation is the function Sign(sk, M) given in Figure 4 in [DILITHIUM-Subm], where sk is the Dilithium private key and M is the data to be signed. OpenPGP does not use the optional randomized signing given as a variant in the definition of this function, i.e. rho' := H(K || mu) is used. The signing function returns the Dilithium signature. That is, to sign with Dilithium the following operation is defined:

```
(dilithiumSignature) <- dilithium.sign(dilithiumPrivateKey,
                                       dataDigest)
```

The procedure for Dilithium signature verification is the function Verify(pk, M, sigma) given in Figure 4 in [DILITHIUM-Subm], where pk is the Dilithium public key, M is the data to be signed and sigma is the Dilithium signature. That is, to verify with Dilithium the following operation is defined:

```
(verified) <- dilithium.verify(dilithiumPublicKey, dataDigest,
                               dilithiumSignature)
```

Dilithium has the parameterization with the corresponding artifact lengths in octets as given in Table 12. All artifacts are encoded as defined in [DILITHIUM-Subm].

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Algorithm ID reference	Dilithium instance	Public key	Secret key	Signature value
35, 37, 39	Dilithium3	1952	4000	3293
36, 38, 40	Dilithium5	2592	4864	4595

Table 12: Dilithium parameters and artifact lengths in octets

6.2. Composite Signature Schemes with Dilithium

6.2.1. Binding hashes

Composite Dilithium + ECC signatures MUST use SHA3-256 (hash algorithm ID 12) or SHA3-512 (hash algorithm ID 14) as hashing algorithm. Signatures using other hash algorithms MUST be considered invalid.

An implementation MUST support SHA3-256 and SHOULD support SHA3-512, in order to support the hash binding with Dilithium + ECC signatures.

6.2.2. Key generation procedure

The implementation MUST independently generate the Dilithium and the ECC component keys. Dilithium key generation follows the specification in [DILITHIUM-Subm] and the artifacts are encoded as fixed-length octet strings as defined in Section 6.1.3. For ECC this is done following the relative specification in [RFC7748], [SP800-186], or [RFC5639], and encoding the artifacts as specified in Section 6.1.1 or Section 6.1.2 as fixed-length octet strings.

6.2.3. Signature Generation

To sign a message M with Dilithium + EdDSA the following sequence of operations has to be performed:

1. Generate dataDigest according to [I-D.ietf-openpgp-crypto-refresh] Section 5.2.4

2. Create the EdDSA signature over dataDigest with eddsa.sign() from Section 6.1.1

3. Create the Dilithium signature over dataDigest with dilithium.sign() from Section 6.1.3

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

4. Encode the EdDSA and Dilithium signatures according to the packet structure given in Section 6.3.1.

To sign a message M with Dilithium + ECDSA the following sequence of operations has to be performed:

1. Generate dataDigest according to [I-D.ietf-openpgp-crypto-refresh] Section 5.2.4

2. Create the ECDSA signature over dataDigest with ecdsa.sign() from Section 6.1.2

3. Create the Dilithium signature over dataDigest with dilithium.sign() from Section 6.1.3

4. Encode the ECDSA and Dilithium signatures according to the packet structure given in Section 6.3.1.

6.2.4. Signature Verification

To verify a Dilithium + EdDSA signature the following sequence of operations has to be performed:

1. Verify the EdDSA signature with eddsa.verify() from Section 6.1.1

2. Verify the Dilithium signature with dilithium.verify() from Section 6.1.3

To verify a Dilithium + ECDSA signature the following sequence of operations has to be performed:

1. Verify the ECDSA signature with ecdsa.verify() from Section 6.1.2

2. Verify the Dilithium signature with dilithium.verify() from Section 6.1.3

As specified in Section 4.3 an implementation MUST validate both signatures, i.e. EdDSA/ECDSA and Dilithium, to state that a composite Dilithium + ECC signature is valid.

6.3. Packet Specifications

6.3.1. Signature Packet (Tag 2)

The composite Dilithium + ECC schemes MUST be used only with v6 signatures, as defined in [I-D.ietf-openpgp-crypto-refresh] Section 5.2.3.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

The algorithm-specific v6 signature parameters for Dilithium + EdDSA signatures consists of:

* A fixed-length octet string representing the EdDSA signature, whose length depends on the algorithm ID as specified in Table 10.

* A fixed-length octet string of the Dilithium signature value, whose length depends on the algorithm ID as specified in Table 12.

The algorithm-specific v6 signature parameters for Dilithium + ECDSA signatures consists of:

* A fixed-length octet string of the big-endian encoded ECDSA value R, whose length depends on the algorithm ID as specified in Table 11.

* A fixed-length octet string of the big-endian encoded ECDSA value S, whose length depends on the algorithm ID as specified in Table 11.

* A fixed-length octet string of the Dilithium signature value, whose length depends on the algorithm ID as specified in Table 12.

6.3.2. Key Material Packets

The composite Dilithium + ECC schemes MUST be used only with v6 keys, as defined in [I-D.ietf-openpgp-crypto-refresh].

The algorithm-specific public key for Dilithium + EdDSA keys is this series of values:

* A fixed-length octet string representing the EdDSA public key, whose length depends on the algorithm ID as specified in Table 10.

* A fixed-length octet string containing the Dilithium public key, whose length depends on the algorithm ID as specified in Table 12.

The algorithm-specific private key for Dilithium + EdDSA keys is this series of values:

* A fixed-length octet string representing the EdDSA secret key, whose length depends on the algorithm ID as specified in Table 10.

* A fixed-length octet string containing the Dilithium secret key, whose length depends on the algorithm ID as specified in Table 12.

The algorithm-specific public key for Dilithium + ECDSA keys is this series of values:

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

* A fixed-length octet string representing the ECDSA public key in SEC1 format, as specified in section Section 2.1.1 and with length specified in Table 11.

* A fixed-length octet string containing the Dilithium public key, whose length depends on the algorithm ID as specified in Table 12.

The algorithm-specific private key for Dilithium + ECDSA keys is this series of values:

* A fixed-length octet string representing the ECDSA secret key as a big-endian encoded integer, whose length depends on the algorithm used as specified in Table 11.

* A fixed-length octet string containing the Dilithium secret key, whose length depends on the algorithm ID as specified in Table 12.

7. SPHINCS+

7.1. The SPHINCS+ Algorithms

The following table describes the SPHINCS+ parameters and artifact lengths:

Parameter ID reference	Parameter name suffix	SPHINCS+ public key	SPHINCS+ secret key	SPHINCS+ signature
1	128s	32	64	7856
2	128f	32	64	17088
3	192s	48	96	16224
4	192f	48	96	35664
5	256s	64	128	29792
6	256f	64	128	49856

Table 13: SPHINCS+ parameters and artifact lengths in octets. The values equally apply to the parameter IDs of SPHINCS+-simple-SHA2 and SPHINCS+-simple-SHAKE.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

7.1.1. Binding hashes

SPHINCS+ signature packets MUST use the associated hash as specified in Table 14. Signature packets using other hashes MUST be considered invalid.

Algorithm ID reference	Parameter ID reference	Hash function	Hash function ID reference
41	1, 2	SHA-256	8
41	3, 4, 5, 6	SHA-512	10
42	1, 2	SHA3-256	12
42	3, 4, 5, 6	SHA3-512	14

Table 14: Binding between SPHINCS+ and signature hashes

An implementation supporting a specific SPHINCS+ algorithm and parameter MUST also support the matching hash algorithm.

7.1.2. Key generation

The SPHINCS+ key generation is performed according to the function spx_keygen() specified in [SPHINCS-Subm], Sec. 6.2 as Alg. 19. The private and public key are encoded as defined in [SPHINCS-Subm].

7.1.3. Signature Generation

The procedure for SPHINCS+ signature generation is the function spx_sign(M, SK) specified in [SPHINCS-Subm], Sec. 6.4 as Alg. 20. Here, M is the dataDigest generated according to [I-D.ietf-openpgp-crypto-refresh] Section 5.2.4 and SK is the SPHINCS+ private key. The global variable RANDOMIZE specified in Alg. 20 is to be considered as not set, i.e. the variable opt shall be initialized with PK.seed. See also Section 9.4.

An implementation MUST set the Parameter ID in the signature equal to the issuing private key Parameter ID.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

7.1.4. Signature Verification

The procedure for SPHINCS+ signature verification is the function spx_verify(M, SIG, PK) specified in [SPHINCS-Subm], Sec. 6.5 as Alg. 21. Here, M is the dataDigest generated according to [I-D.ietf-openpgp-crypto-refresh] Section 5.2.4, SIG is the signature, and PK is the SPHINCS+ public key.

An implementation MUST check that the Parameter ID in the signature and in the key match when verifying.

7.2. Packet specifications

7.2.1. Signature Packet (Tag 2)

The SPHINCS+ algorithms MUST be used only with v6 signatures, as defined in [I-D.ietf-openpgp-crypto-refresh] Section 5.2.3.

The algorithm-specific v6 Signature parameters consists of:

* A one-octet value specifying the SPHINCS+ parameter ID defined in Table 3 and Table 4. The values 0x00 and 0xFF are reserved for future extensions.

* A fixed-length octet string of the SPHINCS+ signature value, whose length depends on the parameter ID in the format specified in Table 13.

7.2.2. Key Material Packets

The SPHINCS+ algorithms MUST be used only with v6 keys, as defined in [I-D.ietf-openpgp-crypto-refresh].

The algorithm-specific public key is this series of values:

* A one-octet value specifying the SPHINCS+ parameter ID defined in Table 3 and Table 4. The values 0x00 and 0xFF are reserved for future extensions.

* A fixed-length octet string containing the SPHINCS+ public key, whose length depends on the parameter ID as specified in Table 13.

The algorithm-specific private key is this value:

* A fixed-length octet string containing the SPHINCS+ secret key, whose length depends on the parameter ID as specified in Table 11.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Wien Bibliothek Your knowledge hub

8. Migration Considerations

The post-quantum KEM algorithms defined in Table 1 and the signature algorithms defined in Table 2 are a set of new public key algorithms that extend the algorithm selection of [I-D.ietf-openpgp-crypto-refresh]. During the transition period, the post-quantum algorithms will not be supported by all clients. Therefore various migration considerations must be taken into account, in particular backwards compatibility to existing implementations that have not yet been updated to support the post-quantum algorithms.

8.1. Key preference

Implementations SHOULD prefer PQ(/T) keys when multiple options are available.

For instance, if encrypting for a recipient for which both a valid PQ/T and a valid ECC certificate are available, the implementation SHOULD choose the PQ/T certificate. In case a certificate has both a PQ/T and an ECC encryption-capable valid subkey, the PQ/T subkey SHOULD be preferred.

An implementation MAY sign with both a PQ(/T) and an ECC key using multiple signatures over the same data as described in Section 4.4. Signing only with PQ(/T) key material is not backwards compatible.

Note that the confidentiality of a message is not post-quantum secure when encrypting to multiple recipients if at least one recipient does not support PQ/T encryption schemes. An implementation SHOULD NOT abort the encryption process in this case to allow for a smooth transition to post-quantum cryptography.

8.2. Key generation strategies

It is REQUIRED to generate fresh secrets when generating PQ(/T) keys. Reusing key material from existing ECC keys in PQ(/T) keys does not provide backwards compatibility, and the fingerprint will differ.

An OpenPGP (v6) certificate is composed of a certification-capable primary key and one or more subkeys for signature, encryption, and authentication. Two migration strategies are recommended:

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

1. Generate two independent certificates, one for PQ(/T)-capable implementations, and one for legacy implementations. Implementations not understanding PQ(/T) certificates can use the legacy certificate, while PQ(/T)-capable implementations will prefer the newer certificate. This allows having an older v4 or v6 ECC certificate for compatibility and a v6 PQ(/T) certificate, at a greater complexity in key distribution.

2. Attach PQ(/T) encryption and signature subkeys to an existing v6 ECC certificate. Implementations understanding PQ(/T) will be able to parse and use the subkeys, while PQ(/T)-incapable implementations can gracefully ignore them. This simplifies key distribution, as only one certificate needs to be communicated and verified, but leaves the primary key vulnerable to quantum computer attacks.

9. Security Considerations

9.1. Hashing in ECC-KEM

Our construction of the ECC-KEMs, in particular the final hashing step in encapsulation and decapsulation that produces the eccKeyShare, is standard and known as hashed ElGamal key encapsulation, a hashed variant of ElGamal encryption. It ensures IND-CCA2 security in the random oracle model under some Diffie-Hellman intractability assumptions [CS03].

9.2. Key combiner

For the key combination in Section 5.2.2 this specification limits itself to the use of KMAC. The sponge construction used by KMAC was proven to be indifferentiable from a random oracle [BDPA08]. This means, that in contrast to SHA2, which uses a Merkle-Damgard construction, no HMAC-based construction is required for key combination. Except for a domain separation it is sufficient to simply process the concatenation of any number of key shares when using a sponge-based construction like KMAC. The construction using KMAC ensures a standardized domain separation. In this case, the processed message is then the concatenation of any number of key shares.

More precisely, for a given capacity c the indifferentiability proof shows that assuming there are no weaknesses found in the Keccak permutation, an attacker has to make an expected number of 2^(c/2) calls to the permutation to tell KMAC from a random oracle. For a random oracle, a difference in only a single bit gives an unrelated, uniformly random output. Hence, to be able to distinguish a key K, derived from shared keys K1 and K2 (and ciphertexts C1 and C2) as

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

```
K = KMAC(domainSeparation, counter || K1 || C1 || K2 || C2 || fixedInfo,
         outputBits, customization)
```

from a random bit string, an adversary has to know (or correctly guess) both key shares K1 and K2, entirely.

The proposed construction in Section 5.2.2 preserves IND-CCA2 of any of its ingredient KEMs, i.e. the newly formed combined KEM is IND-CCA2 secure as long as at least one of the ingredient KEMs is. Indeed, the above stated indifferentiability from a random oracle qualifies Keccak as a split-key pseudorandom function as defined in [GHP18]. That is, Keccak behaves like a random function if at least one input shared secret is picked uniformly at random. Our construction can thus be seen as an instantiation of the IND-CCA2 preserving Example 3 in Figure 1 of [GHP18], up to some reordering of input shared secrets and ciphertexts. In the random oracle setting, the reordering does not influence the arguments in [GHP18].

9.3. Domain separation and binding

The domSeparation information defined in Section 5.2.2 provides the domain separation for the key combiner construction. This ensures that the input keying material is used to generate a KEK for a specific purpose or context.

The fixedInfo defined in Section 5.2.1 binds the derived KEK to the chosen algorithm and communication parties. The algorithm ID identifies univocally the algorithm, the parameters for its instantiation, and the length of all artifacts, including the derived key. The hash of the recipient's public key identifies the subkey used to encrypt the message, binding the KEK to both the Kyber and the ECC key. Given that both algorithms allow a degree of ciphertext malleability, this prevents transformations onto the ciphertext without the final recipient's knowledge.

This is in line with the Recommendation for ECC in section 5.5 of [SP800-56A]. Other fields included in the recommendation are not relevant for the OpenPGP protocol, since the sender is not required to have a key on their own, there are no pre-shared secrets, and all the other parameters are univocally defined by the algorithm ID.

9.4. SPHINCS+

The original specification of SPHINCS+ [SPHINCS-Subm] prescribes an optional randomized hashing. This is not used in this specification, as OpenPGP v6 signatures already provide a salted hash of the appropriate size.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

9.5. Binding hashes in signatures with signature algorithms

In order not to extend the attack surface, we bind the hash algorithm used for message digestion to the hash algorithm used internally by the signature algorithm. Dilithium internally uses a SHAKE256 digest, therefore we require SHA3 in the Dilithium + ECC signature packet. In the case of SPHINCS+ the internal hash algorithm varies based on the algorithm and parameter ID.

10. Additional considerations

10.1. Performance Considerations for SPHINCS+

This specification introduces both Dilithium + ECC as well as SPHINCS+ as PQ(/T) signature schemes.

Generally, it can be said that Dilithium + ECC provides a performance in terms of execution time and space requirements that is close to that of traditional ECC signature schemes. Implementers may want to offer SPHINCS+ for applications where a higher degree of trust in the signature scheme is required. However, SPHINCS+ has performance characteristics in terms of execution time of the signature generation as well as space requirements for the signature that can be, depending on the parameter choice, far greater than those of traditional or Dilithium + ECC signature schemes.

Pertaining to the execution time, the particularly costly operation in SPHINCS+ is the signature generation. In order to achieve short signature generation times, one of the parameter sets with the name ending in the letter "f" for "fast" should be chosen. This comes at the expense of a larger signature size.

In order to minimize the space requirements of a SPHINCS+ signature, a parameter set ending in "s" for "small" should be chosen. This comes at the expense of a larger signature generation time.

11. IANA Considerations

IANA will add the following registries to the Pretty Good Privacy (PGP) registry group at https://www.iana.org/assignments/pgp-parameters:

* Registry name: SPHINCS+-simple-SHA2 parameters

  Registration procedure: SPECIFICATION REQUIRED [RFC8126]

  Values defined in this document, Table 3.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

* Registry name: SPHINCS+-simple-SHAKE parameters

  Registration procedure: SPECIFICATION REQUIRED [RFC8126]

  Values defined in this document, Table 4.

Furthermore IANA will add the algorithm IDs defined in Table 1 and Table 2 to the registry Public Key Algorithms.

12. Contributors

Stephan Ehlen (BSI)
Carl-Daniel Hailfinger (BSI)
Andreas Huelsing (TU Eindhoven)
Johannes Roth (MTG AG)

13. References

13.1. Normative References

[I-D.ietf-openpgp-crypto-refresh]
Wouters, P., Huigens, D., Winter, J., and N. Yutaka, "OpenPGP", Work in Progress, Internet-Draft, draft-ietf-openpgp-crypto-refresh-08, 13 March 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-openpgp-crypto-refresh-08>.

[RFC3394] Schaad, J. and R. Housley, "Advanced Encryption Standard (AES) Key Wrap Algorithm", RFC 3394, DOI 10.17487/RFC3394, September 2002, <https://www.rfc-editor.org/rfc/rfc3394>.

[RFC7748] Langley, A., Hamburg, M., and S. Turner, "Elliptic Curves for Security", RFC 7748, DOI 10.17487/RFC7748, January 2016, <https://www.rfc-editor.org/rfc/rfc7748>.

[RFC8032] Josefsson, S. and I. Liusvaara, "Edwards-Curve Digital Signature Algorithm (EdDSA)", RFC 8032, DOI 10.17487/RFC8032, January 2017, <https://www.rfc-editor.org/rfc/rfc8032>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/rfc/rfc8126>.

13.2. Informative References

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

[BDPA08] Bertoni, G., Daemen, J., Peters, M., and G. Assche, "On the Indifferentiability of the Sponge Construction", 2008, <https://doi.org/10.1007/978-3-540-78967-3_11>.

[CS03] Cramer, R. and V. Shoup, "Design and Analysis of Practical Public-Key Encryption Schemes Secure against Adaptive Chosen Ciphertext Attack", 2003, <https://doi.org/10.1137/S0097539702403773>.

[DILITHIUM-Subm] Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schwabe, P., Seiler, G., and D. Stehle, "CRYSTALS-Dilithium - Algorithm Specifications and Supporting Documentation (Version 3.1)", 8 February 2021.

[draft-driscoll-pqt-hybrid-terminology] Driscoll, F., "Terminology for Post-Quantum Traditional Hybrid Schemes", March 2023, <https://datatracker.ietf.org/doc/html/draft-driscoll-pqt-hybrid-terminology>.

[GHP18] Giacon, F., Heuer, F., and B. Poettering, "KEM Combiners", 2018, <https://doi.org/10.1007/978-3-319-76578-5_7>.

[KYBER-Subm] Avanzi, R., Bos, J., Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schanck, J. M., Schwabe, P., Seiler, G., and D. Stehle, "CRYSTALS-Kyber (version 3.02) - Submission to round 3 of the NIST post-quantum project", 4 August 2021.

[NIST-PQC] Chen, L., Moody, D., and Y. Liu, "Post-Quantum Cryptography Standardization", December 2016, <https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization>.

[NISTIR-8413] Alagic, G., Apon, D., Cooper, D., Dang, Q., Dang, T., Kelsey, J., Lichtinger, J., Miller, C., Moody, D., Peralta, R., Perlner, R., Robinson, A., Smith-Tone, D., and Y. Liu, "Status Report on the Third Round of the NIST Post-Quantum Cryptography Standardization Process", NIST IR 8413 , September 2022, <https://doi.org/10.6028/NIST.IR.8413-upd1>.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

[RFC5639] Lochter, M. and J. Merkle, "Elliptic Curve Cryptography (ECC) Brainpool Standard Curves and Curve Generation", RFC 5639, DOI 10.17487/RFC5639, March 2010, <https://www.rfc-editor.org/rfc/rfc5639>.

[SEC1] Standards for Efficient Cryptography Group, "Standards for Efficient Cryptography 1 (SEC 1)", May 2009, <https://secg.org/sec1-v2.pdf>.

[SP800-185]
Kelsey, J., Chang, S., and R. Perlner, "SHA-3 Derived Functions: cSHAKE, KMAC, TupleHash, and ParallelHash", NIST Special Publication 800-185 , December 2016, <https://doi.org/10.6028/NIST.SP.800-185>.

[SP800-186]
Chen, L., Moody, D., Regenscheid, A., and K. Randall, "Recommendations for Discrete Logarithm-Based Cryptography: Elliptic Curve Domain Parameters", NIST Special Publication 800-186 , February 2023, <https://doi.org/10.6028/NIST.SP.800-186>.

[SP800-56A]
Barker, E., Chen, L., Roginsky, A., Vassilev, A., and R. Davis, "Recommendation for Pair-Wise Key-Establishment Schemes Using Discrete Logarithm Cryptography", NIST Special Publication 800-56A Rev. 3 , April 2018, <https://doi.org/10.6028/NIST.SP.800-56Ar3>.

[SP800-56C]
Barker, E., Chen, L., and R. Davis, "Recommendation for Key-Derivation Methods in Key-Establishment Schemes", NIST Special Publication 800-56C Rev. 2 , August 2020, <https://doi.org/10.6028/NIST.SP.800-56Cr2>.

[SPHINCS-Subm]
Aumasson, J., Bernstein, D. J., Beullens, W., Dobraunig, C., Eichlseder, M., Fluhrer, S., Gazdag, S., Huelsing, A., Kampanakis, P., Koelb, S., Lange, T., Lauridsen, M. M., Mendel, F., Niederhagen, R., Rechberger, C., Rijneveld, J., Schwabe, P., and B. Westerbaan, "SPHINCS+ - Submission to the 3rd round of the NIST post-quantum project. v3.1", 10 June 2021.

Acknowledgments

Thanks to Daniel Huigens and Evangelos Karatsiolis for the early review and feedback on this document.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

```
Kousidis, et al.       Expires 26 September 2023              [Page 38]

Internet-Draft              PQC in OpenPGP                   March 2023
```


```
Authors' Addresses

   Stavros Kousidis
   BSI
   Germany
   Email: stavros.kousidis@bsi.bund.de


   Falko Strenzke
   MTG AG
   Germany
   Email: falko.strenzke@mtg.de


   Aron Wussler
   Proton AG
   Switzerland
   Email: aron@wussler.it
```


Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

List of Figures

2.1 Version 6 of an X25519 PKESK packet structure. 9
2.2 Version 6 Ed25519 signature packet structure. 10
2.3 Version 6 SKESK packet structure. 11
2.4 Version 6 OPS signature packet structure. 12
2.5 Version 6 public and secret key packets. The secret key is a superset of the public key, containing all of its parameters, plus the optionally encrypted algorithm-specific secret key material. 12
2.6 Literal data packet structure. 13
2.7 Version 2 SEIPD packet structure. 14
2.8 Random sponge function construction. 23

3.1 Composite vs Composable PKESK packet structure. Changes highlighted in yellow. 32
3.2 Composite vs Composable key packet structure. Changes highlighted in yellow. 33
3.3 Composite signature packet. Changes highlighted in yellow. 39

5.1 Overview of the artifact size for all newly implemented and reference algorithms for 1 KB of data. The plot is logarithmic in the size of the artifact. 53
5.2 Overview of the x86 performance for all newly implemented and reference algorithms for 1 KB of data. The plot is logarithmic in the operation time. 55
5.3 Overview of the ARM performance for all newly implemented and reference algorithms for 1 KB of data. The plot is logarithmic in the operation time. 56
5.4 Time comparison for key generation of compact SPHINCS$^+$ vs RSA . . . 58
5.5 Time comparison for key generation of fast SPHINCS$^+$ vs RSA 59
5.6 Time comparison for key generation of Dilithium vs NIST and CFRG curves 60
5.7 Time comparison for key generation of Dilithium vs Brainpool curves . . 61
5.8 Time comparison of key parsing on x86 62
5.9 Time comparison of key parsing on ARM 63
5.10 Time comparison for signature generation of the compact variant of SPHINCS$^+$ vs RSA 64
5.11 Time comparison for signature generation of the fast variant of SPHINCS$^+$ vs RSA 65
5.12 Time comparison of signature generation on x86 for Dilithium vs EC . . . 66

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

5.13 Time comparison of signature generation on ARM for Dilithium vs EC . . 67
5.14 Time comparison of verification on x86 68
5.15 Time comparison of verification on ARM 69
5.16 Time comparison of encryption 70
5.17 Time comparison of decryption 72

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

List of Tables

3.1 Proposed KEM algorithm IDs and their implementation requirement. . . 34
3.2 Proposed signature algorithm IDs and their implementation requirement. 38
3.3 Proposed SPHINCS$^+$ parametrization. 38
3.4 Proposed signature hash binding. 40

5.1 Default matching for public and private keys in go-crypto 52

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

TU WIEN Bibliothek Your knowledge hub
Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Acronyms

AEAD Authenticated Encryption with Associated Data. 11, 12, 14, 41

AES Advanced Encryption Standard. 37

BSI German Federal Office for Information Security. 1, 2, 7, 29

CCA Chosen Ciphertext Attack. 22, 35

CFRG Crypto Forum Research Group. 1, 7, 20, 60, 117

DH Diffie-Hellman. 2

DSA Digital Signature Algorithm. 2, 40

EC Elliptic Curve. 25, 37, 38, 40, 44, 46, 49, 52, 60, 61, 66, 67, 74, 117, 118

ECDH Elliptic Curve Diffie-Hellman. 2, 25, 31, 37, 40, 44, 45, 49

ECDSA Elliptic Curve Digital Signature Algorithm. 2, 37, 40, 44–47, 73

EdDSA Edwards-curve Digital Signature Algorithm. 2, 37, 40, 44–47, 73

FIPS Federal Information Processing Standard. 7, 17

FORS Forest Of Random Subsets. 19

FPGA Field Programmable Gate Array. 7, 18

HKDF HMAC-based Key Derivation Function. 14

HMAC Hash-based Message Authentication Code. 36

HSM Hardware Security Module. 7

HT Hyper-Tree. 19

HTTPS Hypertext Transfer Protocol Secure. 7, 8

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

IETF Internet Engineering Task Force. 1, 29, 49, 74, 77

IND–CCA2 Indistinguishability under adaptive Chosen Ciphertext Attack. 21, 22, 25, 34–36, 49, 73

IRTF Internet Research Task Force. 21

IV Initialization Vector. 11, 12, 14

KDF Key Derivation Function. 21, 25, 27, 30, 35–37, 46, 49, 73

KEK Key Encryption Key. 9, 11, 22, 35–37

KEM Key Encapsulation Mechanism. 6, 15–17, 20–22, 25, 30, 34, 36, 37, 52, 73, 75, 119

KMAC Keccak Message Authentication Code. 21, 24–27, 35–37, 73

LMS Leighton-Micali Signatures. 18

LWE Learning With Errors. 16, 17

MAC Message Authentication Code. 24

MIME Multipurpose Internet Mail Extensions. 8

MLWE Module Learning With Errors. 16–18

MPI Multi-Precision Integer. 40

NIST National Institute of Standards and Technology. 1, 2, 5, 7, 15–18, 20, 21, 27, 36, 40, 45, 49, 51, 60, 73, 117

NSA National Security Agency. 7, 49

OPS One-Pass Signature. 11, 12, 15, 40, 41, 117

OTS One-Time Signature. 18

PGP Pretty Good Privacy. 5

PKE Public-Key Encryption. 9, 17, 30, 38, 49

PKESK Public-Key Encrypted Session Key. 9, 15, 30, 32, 34, 37, 40, 41, 49, 117

PKS Public Key Server. 7

PQ Post-Quantum. xi, xiii, 1–3, 5, 11, 13, 15, 16, 25, 29, 31, 33, 35, 37–41, 44–46, 48, 49, 51, 52, 54, 57, 73–75

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

PRF Pseudo-Random Function. 19, 24, 26, 39

QC-MDPC Quasi-Cyclic Moderate Density Parity Check. 20

RFC Request For Comments. 2, 5, 18, 21, 37, 73–75, 77

RSA Rivest-Shamir-Adleman. 2, 40, 52, 57–59, 64–66, 74, 117

S2K String to Key. 11, 12

SEIPD Symmetrically Encrypted Integrity Protected Data. 14, 15, 34, 41, 117

SHA–3 Secure Hash Algorithm 3. 21, 35–37, 39, 41

SKESK Symmetric-Key Encrypted Session Key. 10, 11, 15, 117

SKPRF Split-Key Pseudo-Random Function. 25–27

SOP Stateless OpenPGP Protocol. 31

SSH Secure Shell Protocol. 1, 3, 5, 6

SVP Shortest Vector Problem. 2, 16

TLS Transport Layer Security. 1, 3, 5, 6

WG Working Group. 75

WKD Web Key Directory. 6, 7

WOTS+ Winternitz One Time Signature. 19

XMSS Extended Merkle Signature Scheme. 18, 19

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Bibliography

[AAB+19] Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C. Bardin, Rami Barends, Rupak Biswas, Sergio Boixo, Fernando G. S. L. Brandao, David A. Buell, Brian Burkett, Yu Chen, Zijun Chen, Ben Chiaro, Roberto Collins, William Courtney, Andrew Dunsworth, Edward Farhi, Brooks Foxen, Austin Fowler, Craig Gidney, Marissa Giustina, Rob Graff, Keith Guerin, Steve Habegger, Matthew P. Harrigan, Michael J. Hartmann, Alan Ho, Markus Hoffmann, Trent Huang, Travis S. Humble, Sergei V. Isakov, Evan Jeffrey, Zhang Jiang, Dvir Kafri, Kostyantyn Kechedzhi, Julian Kelly, Paul V. Klimov, Sergey Knysh, Alexander Korotkov, Fedor Kostritsa, David Landhuis, Mike Lindmark, Erik Lucero, Dmitry Lyakh, Salvatore Mandrà, Jarrod R. McClean, Matthew McEwen, Anthony Megrant, Xiao Mi, Kristel Michielsen, Masoud Mohseni, Josh Mutus, Ofer Naaman, Matthew Neeley, Charles Neill, Murphy Yuezhen Niu, Eric Ostby, Andre Petukhov, John C. Platt, Chris Quintana, Eleanor G. Rieffel, Pedram Roushan, Nicholas C. Rubin, Daniel Sank, Kevin J. Satzinger, Vadim Smelyanskiy, Kevin J. Sung, Matthew D. Trevithick, Amit Vainsencher, Benjamin Villalonga, Theodore White, Z. Jamie Yao, Ping Yeh, Adam Zalcman, Hartmut Neven, and John M. Martinis. Quantum supremacy using a programmable superconducting processor. *Nature*, 574(7779):505–510, October 2019.

[ABB+22a] Nicolas Aragon, Paulo S.L.M. Barreto Barreto, Slim Bettaieb, Loic Bidoux, Olivier Blazy, Jean-Christophe Deneuville, Phillipe Gaborit, Shay Gueron, Tim Güneysu, Carlos Aguilar Melchor, Rafael Misoczki, Edoardo Persichetti, Jan Richter-Brockmann, Nicolas Sendrier, Jean-Pierre Tillich, Valentin Vasseur, and Gilles Zemor. Bike: bit flipping key encapsulation. Technical report, 2022.

[ABB+22b] Jean-Philippe Aumasson, Daniel J. Bernstein, Ward Beullens, Christoph Dobraunig, Maria Eichlseder, Scott Fluhrer, Stefan-Lukas Gazdag, Andreas Hülsing, Panos Kampanakis, Stefan Kölbl, Tanja Lange, Martin M. Lauridsen, Florian Mendel, Ruben Niederhagen, Christian Rechberger, Joost Rijneveld, Peter Schwabe, and Bas Westerbaan. Sphincs+ submission to the nist post-quantum project, v.3.1, June 2022.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

[ABC+22] Martin R. Albrecht, Daniel J. Bernstein, Tung Chou, Carlos Cid, Jan Gilcher, Tanja Lange, Varun Maram, Ingo von Maurich, Rafael Misoczki, Ruben Niederhagen, Kenneth G. Paterson, Edoardo Persichetti, Christiane Peters, Peter Schwabe, Nicolas Sendrier, Jakub Szefer, Cen Jung Tjhai, Martin Tomlinson, and Wen Wang. Classic mceliece: conservative code-based cryptography: cryptosystem specification, 2022.

[ABD+21] Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé. Crystals-kyber (version 3.02) – submission to round 3 of the nist post-quantum project, August 2021.

[ADK+22] Nimrod Aviram, Benjamin Dowling, Ilan Komargodski, Kenneth G. Paterson, Eyal Ronen, and Eylon Yogev. Practical (post-quantum) key combiners from one-wayness and applications to tls. Cryptology ePrint Archive, Paper 2022/065, 2022. `https://eprint.iacr.org/2022/065`.

[Age22] National Security Agency. The commercial national security algorithm suite 2.0 and quantum computing faq, 2022.

[Ajt96] M. Ajtai. Generating hard instances of lattice problems (extended abstract). In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing - STOC '96*. ACM Press, 1996.

[BCD20] Elaine Barker, Lily Chen, and Richard Davis. Recommendation for key-derivation methods in key-establishment schemes. Technical report, August 2020.

[BDKJ21] Alex Biryukov, Daniel Dinu, Dmitry Khovratovich, and Simon Josefsson. Argon2 Memory-Hard Function for Password Hashing and Proof-of-Work Applications. RFC 9106, September 2021.

[BDPvA11] Guido Bertoni, Joan Daemen, Michael Peters, and Gilles van Assche. Cryptographic sponge functions, January 2011.

[BHP22] Lara Bruseghini, Daniel Huigens, and Kenneth G. Paterson. Victory by ko: Attacking openpgp using key overwriting. *Proceedings of ACM Conference on Computer and Communications Security, Los Angeles*, 2022.

[BL17] Daniel J. Bernstein and Tanja Lange. Post-quantum cryptography. *Nature*, 549(7671):188–194, September 2017.

[BT120] Deutscher bundestag: Antwort der bundesregierung auf die kleine anfrage der abgeordneten manuel höferlin, frank sitta, grigorios aggelidis, weiterer abgeordneter und der fraktion der fdp – drucksache 19/17500 – hochsicheres quantennetzwerk qunet. March 2020.

[Cou22] NIST General Counsel. Nist pqc license summary and excerpts, 2022.

[CS03] Ronald Cramer and Victor Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM Journal on Computing*, 33(1):167–226, 2003.

[DKL+21] Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. Crystals-dilithium – algorithm specifications and supporting documentation (version 3.1), February 2021.

[EHH+20] Stephan Ehlen, Heike Hagemeier, Tobias Hemmert, Stavros Kousidis, Manfred Lochter, Stephanie Reinhardt, and Thomas Wunderer. Quantum-safe cryptography - fundamentals, current developments and recommendations, 2020.

[EHK+22] Stephan Ehlen, Andreas Hülsing, Evangelos Karatsiolis, Stavros Kousidis, Johannes Roth, Falko Strenzke, Christian Tobias, and Aron Wussler. Standardization efforts for pqc in openpgp in the project pqc@thunderbird, 2022.

[ELPa] Karen Easterbrook, Brian LaMacchia, and Christian Paquin. Post-quantum tls.

[ELPb] Karen Easterbrook, Brian LaMacchia, and Christian Paquin. Post-quantum tls.

[FDC+07] Hal Finney, Lutz Donnerhacke, Jon Callas, Rodney L. Thayer, and David Shaw. OpenPGP Message Format. RFC 4880, November 2007.

[FHK+20] Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Prest, Thomas Ricosset, Gregor Seiler, William Whyte, and Zhenfei Zhang. Falcon: Fast-fourier lattice-based compact signatures over ntru, 2020.

[fIS20] Federal Office for Information Security. Migration to post quantum cryptography, 2020.

[FT14] S. Farrell and H. Tschofenig. Pervasive monitoring is an attack. BCP 188, RFC Editor, May 2014. `http://www.rfc-editor.org/rfc/rfc7258.txt`.

[FTDC98] Hal Finney, Rodney L. Thayer, Lutz Donnerhacke, and Jon Callas. OpenPGP Message Format. RFC 2440, November 1998.

[GGH96] Oded Goldreich, Shafi Goldwasser, and Shai Halevi. Public-key cryptosystems from lattice reduction problems. Cryptology ePrint Archive, Report 1996/016, 1996.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

[GHP18] Federico Giacon, Felix Heuer, and Bertram Poettering. Kem combiners. Cryptology ePrint Archive, Paper 2018/024, 2018.

[Gil22] Daniel Kahn Gillmor. Stateless OpenPGP Command Line Interface. Internet-Draft draft-dkg-openpgp-stateless-cli-04, Internet Engineering Task Force, May 2022. Work in Progress.

[HBG+18] Andreas Huelsing, Denis Butin, Stefan-Lukas Gazdag, Joost Rijneveld, and Aziz Mohaisen. XMSS: eXtended Merkle Signature Scheme. RFC 8391, May 2018.

[HMOR21] James Howe, Marco Martinoli, Elisabeth Oswald, and Francesco Regazzoni. Exploring parallelism to improve the performance of frodokem in hardware. Cryptology ePrint Archive, Report 2021/155, 2021.

[Hof20] Paul E. Hoffman. The Transition from Classical to Post-Quantum Cryptography. Internet-Draft draft-hoffman-c2pq-07, Internet Engineering Task Force, May 2020. Work in Progress.

[HS02] Russ Housley and Jim Schaad. Advanced Encryption Standard (AES) Key Wrap Algorithm. RFC 3394, October 2002.

[Jiv12] Andrey Jivsov. Elliptic Curve Cryptography (ECC) in OpenPGP. RFC 6637, June 2012.

[KE10] Dr. Hugo Krawczyk and Pasi Eronen. HMAC-based Extract-and-Expand Key Derivation Function (HKDF). RFC 5869, May 2010.

[KjCP16] John Kelsey, Shu jen Chang, and Ray Perlner. Sha-3 derived functions: cshake, kmac, tuplehash, and parallelhash. Technical report, August 2016.

[KL20] J. Katz and Y. Lindell. *Introduction to Modern Cryptography*. Chapman & Hall/CRC Cryptography and Network Security Series. CRC Press, 2020.

[Koc16] Werner Koch. EdDSA for OpenPGP. Internet-Draft draft-koch-eddsa-for-openpgp-04, Internet Engineering Task Force, February 2016. Work in Progress.

[Koc22] Werner Koch. OpenPGP Web Key Directory. Internet-Draft draft-koch-openpgp-webkey-service-15, Internet Engineering Task Force, November 2022. Work in Progress.

[KSL+19] Krzysztof Kwiatkowski, Nick Sullivan, Adam Langley, Dave Levin, and Alan Mislove. Measuring tls key exchange with post-quantum kem, 2019.

[KV] Krzysztof Kwiatkowski and Luke Valenta. The tls post-quantum experiment.

[Lam79] Leslie Lamport. Constructing digital signatures from a one way function. Technical Report CSL-98, October 1979.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

[LHT16] Adam Langley, Mike Hamburg, and Sean Turner. Elliptic Curves for Security. RFC 7748, January 2016.

[LP20] Gaëtan Leurent and Thomas Peyrin. Sha-1 is a shambles - first chosen-prefix collision on sha-1 and application to the pgp web of trust. Cryptology ePrint Archive, Paper 2020/014, 2020. `https://eprint.iacr.org/2020/014`.

[LS12] Adeline Langlois and Damien Stehle. Worst-case to average-case reductions for module lattices. Cryptology ePrint Archive, Paper 2012/090, 2012.

[MAA+22] Dustin Moody, Gorjan Alagic, Daniel C Apon, David A Cooper, Quynh H Dang, Thinh Dang, John M Kelsey, Jacob Lichtinger, Yi-Kai Liu, Carl A Miller, Rene C Peralta, Ray A Perlner, Angela Y Robinson, and Daniel C Smith-Tone. Status report on the third round of the NIST post-quantum cryptography standardization process. Technical report, July 2022.

[McE78] R J McEliece. A public-key cryptosystem based on algebraic coding theory. In *DSN Prog. Rep., Jet Prop. Lab., California Inst. Technol*, pages 114–116, 1978.

[MCF19] David McGrew, Michael Curcio, and Scott Fluhrer. Leighton-Micali Hash-Based Signatures. RFC 8554, April 2019.

[Mer79] Ralph Merkle. Secrecy, authentication, and public key systems. Technical report, 1979.

[MI88] Tsutomu Matsumoto and Hideki Imai. Public quadratic polynomial-tuples for efficient signature-verification and message-encryption. In *Lecture Notes in Computer Science*, pages 419–453. Springer Berlin Heidelberg, 1988.

[ML10] Johannes Merkle and Manfred Lochter. Elliptic Curve Cryptography (ECC) Brainpool Standard Curves and Curve Generation. RFC 5639, March 2010.

[n.A17] n.A. Ballot 193 – 825-day certificate lifetimes, 2017.

[OWK23] Mike Ounsworth, Aron Wussler, and Stavros Kousidis. Combiner function for hybrid key encapsulation mechanisms (Hybrid KEMs). Internet-Draft draft-ounsworth-cfrg-kem-combiners-03, Internet Engineering Task Force, March 2023. Work in Progress.

[PDM+18] Damian Poddebniak, Christian Dresen, Jens Müller, Fabian Ising, Sebastian Schinzel, Simon Friedberger, Juraj Somorovsky, and Jörg Schwenk. Efail: Breaking S/MIME and OpenPGP email encryption using exfiltration channels. In *27th USENIX Security Symposium (USENIX Security 18)*, pages 549–566, Baltimore, MD, August 2018. USENIX Association.

[Res01] Pete Resnick. Internet Message Format. RFC 2822, April 2001.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

[RMJ+21] Sara Ricci, Lukas Malina, Petr Jedlicka, David Smekal, Jan Hajny, Petr Cibik, and Patrik Dobias. Implementing crystals-dilithium signature scheme on fpgas. Cryptology ePrint Archive, Paper 2021/108, 2021.

[Sho94] P.W. Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th Annual Symposium on Foundations of Computer Science*. IEEE Comput. Soc. Press, 1994.

[SKD20] Dimitrios Sikeridis, Panos Kampanakis, and Michael Devetsikiotis. *Assessing the Overhead of Post-Quantum Cryptography in TLS 1.3 and SSH*, page 149–156. Association for Computing Machinery, New York, NY, USA, 2020.

[WHWY23] Paul Wouters, Daniel Huigens, Justus Winter, and Niibe Yutaka. OpenPGP. Internet-Draft draft-ietf-openpgp-crypto-refresh-08, Internet Engineering Task Force, March 2023. Work in Progress.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub