Post-Quantum cryptography in OpenPGP

NIST Fifth PQC Standardization Conference

Aron Wussler[1]

Proton AG, aron@wussler.it

1 Introduction

This paper reports on a project that 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 TLS [KSL+19,KV,ELPa] or SSH [ELPb,SKD20] have been experimenting with PQ schemes to determine any performance drawback or issue with the larger artifacts.

The project has so far produced a draft standard proposal [KSW23] for the use of PQC in OpenPGP, which is standardized at the Internet Engineering Task Force (IETF).

The criteria chosen in this project were to use already standardized algorithms, mostly relying on the 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 standard proposal, 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.

To evaluate the results of the project, the draft standard was implemented in the `go-crypto` Golang library and benchmarked.

2 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].

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 $x + y > z$.

Given that standardization is a lengthy and complex process, than OpenPGP is an asynchronous protocol where changes can take a long time to propagate, and 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.

3 OpenPGP

OpenPGP is a protocol mostly used to perform E2EE communication, encrypted data storage, software signing, and authentication. Most of its uses are asynchronous and key distribution is generally off-band. This project has been designed as an extension of its latest version [WHWY24], introducing new code points for the PQ algorithms.

It is important to note that OpenPGP presents some significant differences from other protocols where PQ algorithms were were experimented, such as TLS or OpenSSH, because of its asynchronous context. Removing real-time requirements implies some advantages, enabling larger artifacts and slower processing times, and some disadvantages, namely the legacy burden. Artifacts, such as ciphertexts and signatures may be accessed years after their creation, and therefore libraries must keep long term support for deprecated algorithms.

OpenPGP is also being used on secure enclaves and consumer Hardware Security Modules (HSMs), thus it is important that the used algorithms will be available on said devices. Finally, OpenPGP is a critical component of Linux software signing, where hash-based signatures are a welcome feature.

4 Algorithm selection

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.

In the PQ algorithm selection process we decided to evaluate only algorithms undergoing a standardization process, because of OpenPGP's legacy burden: once a scheme is deployed, implementations have to support it for a virtually unbounded time, and offering a non-standard algorithm complicates future implementations, for instance by having poor access to certified libraries or security fixes.

We excluded all the stateful schemes, because OpenPGP keys are often cloned across devices and provide very poor support for state synchronization. We therefore selected SLH-DSA because offering a stateless hash-based scheme is very interesting, as it provides long-term security guarantees for software signing. OpenPGP can bear the expense of greater bandwidth usage and slower signature generation.

Structured lattice and code-based algorithms are also interesting, since they provide performance similar to the existing algorithms.

We selected ML-KEM because it offers good performance, comparable to elliptic curves, and artifact size in the 1 to 3 KB range for security level 3 and 5. We took in consideration that NIST believes that Module Learning With Errors (MLWE) is suitable for high-performance cryptosystems without sacrificing security, and the ML-KEM team provides an extensive security analysis with concrete estimates for the security parameters. It is therefore the only Key Encapsulation Mechanism (KEM) candidate chosen for standardization, eventually giving it Federal Information Processing Standard (FIPS) compliance and setting it up for wide adoption with solid and reviewed implementations. ML-KEM has some known patent issues, but in November 2022 the intellectual property agreement was released [Cou22], stating that adopting the NIST standardized version of ML-KEM is going to be royalty-free.

We also selected ML-DSA because it 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 ML-KEM, and has already been successfully implemented on Field Programmable Gate Array (FPGA) circuits [RMJ+21].

For these two structured lattice based schemes, as recommended from the BSI [EHH+20] as well as the authors of CRYSTALS-Kyber and CRYSTALS-Dilithium, we opted for introducing hybrid algorithms for ML-KEM and ML-DSA. Given that OpenPGP is a protocol with very little crypto-agility, where long-lived keys are very common, combining the security of both the elliptic curves and PQ was deemed a sound decision by both the authors and the community; ensuring that any flaw in the algorithm or implementation does not compromise data security.

We excluded Falcon [FHK+20] because it requires floating-point arithmetics, therefore making implementations complex, especially if constant-time operations are desired. In comparison to the other candidates, it features compact signatures, but given that bandwidth is not a critical requirement for OpenPGP, ML-DSA appears as a better lattice-based candidate. To reduce the legacy burden we decided to limit the number of lattice-based candidates to one.

We also excluded Classic McEliece [ABC+22], because to achieve PQ safe 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. It is important to consider that OpenPGP certificates can contain multiple subkeys, and this may easily generate certificates in the tens of megabytes. For instance, a typical keyserver dataset of 6 million keys currently fits into 80GB at runtime, while it would shift to terabytes if McEliece keys became commonplace.

Finally we excluded BIKE and HQC because there is no active standardization process. If a standardization body were to initiate the process, it could be a great candidate for an alternative KEM based on a different underlying problem.

4.1 Composable vs Composite

When implementing hybrid schemes we opted for composite schemes, designed to have a fixed combination of two or more algorithms, coming as 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. We encountered no difficulty

implementing this interface into the existing protocol, as we could reuse the Elliptic Curve Integrated Encryption Scheme (ECIES) KEM construction. 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 key combiner construction.

While composable schemes allow more crypto-agility, as creating a new combination does not require standardization of a new identifier or updating the deployed software, we still preferred composite schemes because they are easy to identify, provide a simple framework to establish a security policy, reduce overall complexity, and improve interoperability.

Furthermore, choosing composite schemes prevented from creating backwards-compatible keys and artifacts. While this may be seen as a user-hostile choice, we argue that key distribution will anyway be necessary for the PQ keys, and users can choose to distribute both PQ and traditional keys (see section 7). This approach significantly reduces the attack surface for downgrade attacks and lets the key holder decide what is their policy regarding PQ messages.

Finally, it is to be considered that 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.

4.2 Algorithm list

To encrypt messages, we selected to use the following KEMs: ML-KEM 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. Conforming with the existing specification, X25519 has been chosen as mandatory to implement, X448 as recommended, and the remaining are optional; this to fulfill specific compliance requirements and compatibility with existing hardware.

The secrets derived from ML-KEM and the curve are then combined using a Keccak Message Authentication Code (KMAC) based key combiner compliant with NIST SP800-56C [BCD20] that preserves the Indistinguishability under adaptive Chosen Ciphertext Attack (IND-CCA2) security of each KEM.

To sign messages, we selected ML-DSA hybrid with Ed25519, Ed448, P-256, P-384, Brainpool P-256, or Brainpool P-384; the first two curves with the Edwards-curve Digital Signature Algorithm (EdDSA), while the latter four with the Elliptic Curve Digital Signature Algorithm (ECDSA). For the same reasons as ML-KEM, Ed25519 has been chosen as mandatory to implement, Ed448 as recommended, and the remaining are optional.

We also selected SLH-DSA as standalone, as the confidence in its proof and the security of the underlying hashes is very high. Both the SHA-2 and SHAKE variants are available, the former is recommended, the second optional, because the SHA-2 variant offers significantly better performance while retaining the security guarantees.

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. SLH-DSA 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 trade-off made with SLH-DSA we decided to parameterize it, allowing implementations and users to determine how critical is speed in their usage.

5 Key Combination

To achieve hybrid encryption, a Key Derivation Function (KDF) derives a Key Encryption Key (KEK) from the traditional and PQ key shares. We opted for a generic KMAC-based construction, compliant with NIST SP800-56C [BCD20]. Featuring a generic construction allows to extend the specification with further KEMs featuring fixed-length key shares without having to implement a second key combiner. As long as one of the components provides IND-CCA2 security, the overall construction is also IND-CCA2 secure. This property is proven in 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). Assuming there are no weaknesses found in the Keccak permutation it can be shown that KMAC is an SKPRF, therefore our construction is equivalent to said example.

The following KMAC-based design is used:

```
//   multiKeyCombine(eccKeyShare, eccCipherText,
```

```
//                   mlkemKeyShare, mlkemCipherText,
//                   fixedInfo, oBits)
//
//   Input:
//   eccKeyShare     - the ECC key share encoded as an octet string
//   eccCipherText   - the ECC ciphertext encoded as an octet string
//   mlkemKeyShare   - the ML-KEM key share encoded as an octet string
//   mlkemCipherText - the ML-KEM 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"

eccData = eccKeyShare || eccCipherText
mlkemData = mlkemKeyShare || mlkemCipherText
encData = counter || eccData || mlkemData || fixedInfo

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

The Internet Research Task Force (IRTF) draft `draft-ounsworth-cfrg-kem-combiners-03` [OWK23] was derived from a generalization of this construction, and is proposed as a protocol-agnostic key combiner.

6 Salted Hashes

OpenPGP uses a hash-then-sign paradigm on the protocol level, in order to implement easier data streaming. In particular, there is a One-Pass Signature (OPS) packet preceding the data, informing that a signature packet is at the end of the data stream. Version 6 OPS include an unpredictable salt that is prepended to the hashed data, which size depends on the hashing algorithm. This makes v6 OpenPGP signatures non-deterministic and protects against a broad class of attacks that depend on creating a signature over a predictable message [LP20]. This feature ensures that target-collision resistance is sufficient for the hash function used, similar to the signature randomization mechanism of SLH-DSA.

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: implementations might not have access to the data multiple times to rewind after generating the pseudorandom salt from the message.

Considered that PQ algorithms are bound to use version 6 signatures, featuring salted hashing, the PQ draft specification proposes to keep the hash-then-sign paradigm. The internal SLH-DSA and the OpenPGP salt lengths are required to match.

7 Migration strategies

We propose two different migration strategies, depending on the use case. The first strategy is based on having two independent 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 add to the specification a “key superseded” signature, that allows to designate a replacement key without effectively revoking the old one.

The second strategy consists of attaching a PQ encryption subkey to a traditional 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.

8 Performance Analysis

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 benchmarked to compare it with the existing algorithms on desktop devices and Android phones, compiling it via gomobile.

To measure the performance of the algorithms and implementation we timed end-to-end key generation, parsing, encryption, signing, and verification of round 3 submissions of ML-KEM, ML-DSA and SLH-DSA.

Our objective with these measurement is to provide a realistic user perspective and reason about the user experience. 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].

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, and the encryption subkey via a binding signature.

Therefore, key generation operation consists of two different primitive key generations and three signature generations; key parsing includes three signature verifications; while encryption, decryption, signing and verification are just the primitive operations over 1 KB of randomly generated plaintext.

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 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
ML-DSA-65 + Ed25519	ML-KEM-768 + X25519
ML-DSA-87 + Ed448	ML-KEM-1024 + X448
ML-DSA-65 + ECDSA-NIST-P-256	ML-KEM-768 + ECDH-NIST-P-256
ML-DSA-87 + ECDSA-NIST-P-384	ML-KEM-1024 + ECDH-NIST-P-384
ML-DSA-65 + ECDSA-brainpoolP256r1	ML-KEM-768 + ECDH-brainpoolP256r1
ML-DSA-87 + ECDSA-brainpoolP384r1	ML-KEM-1024 + ECDH-brainpoolP384r1
SLH-DSA–simple–SHA2	ML-KEM-1024 + X448
SLH-DSA–simple–SHAKE	ML-KEM-1024 + X448

Table 1. Default matching for public and private keys in go-crypto

8.1 Artifact Size

The artifact size for each considered algorithm combination and operation is shown in fig. 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 SLH-DSA 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. This trade-off 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. 1 refer only to the algorithm overhead, i.e. excluding the signature plaintext and the encrypted ciphertext. For in-

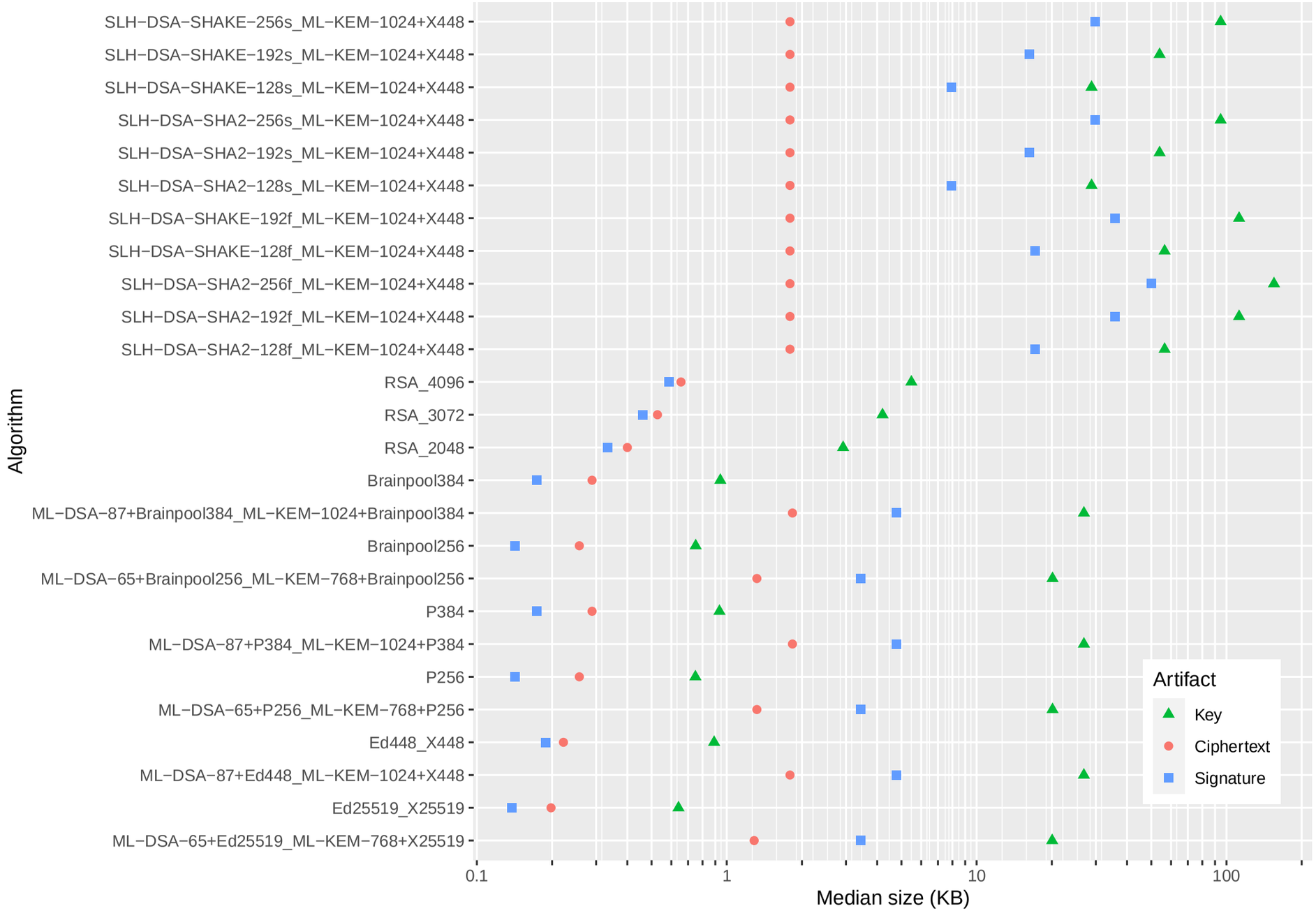

Fig. 1. Overview of the artifact size for all newly implemented and reference algorithms for 1 KB of data using a round 3 submission implementation. The plot is logarithmic in the size of the artifact.

stance, for a 1KB large email, X25519 will add 0.2KB of overhead totalling 1.2KB message size, while ML-KEM-768 combined with X25519 will add 2KB of overhead, totalling 3KB.

8.2 Desktop Performance Analysis

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 was chosen to ensure realistic measurements from the user perspective.

We can observe an overview of the median performance of all considered algorithms in fig. 2. Here 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 trade-offs between security, size, and speed.

8.3 Mobile Performance Analysis

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. 3.

9 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 SLH-DSA, 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' performance 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, SLH-DSA 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 mandatory algorithms, the implementation has a similar performance profile to today's operations, allowing for a transparent migration from the user's perspective.

10 Directions for Future Research

10.1 OpenPGP 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 the upcoming meetings, where the draft has been proposed for adoption.

10.2 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 structured lattices. In particular, code theory based algorithms such as BIKE or HQC are considered very interesting, if NIST 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.

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

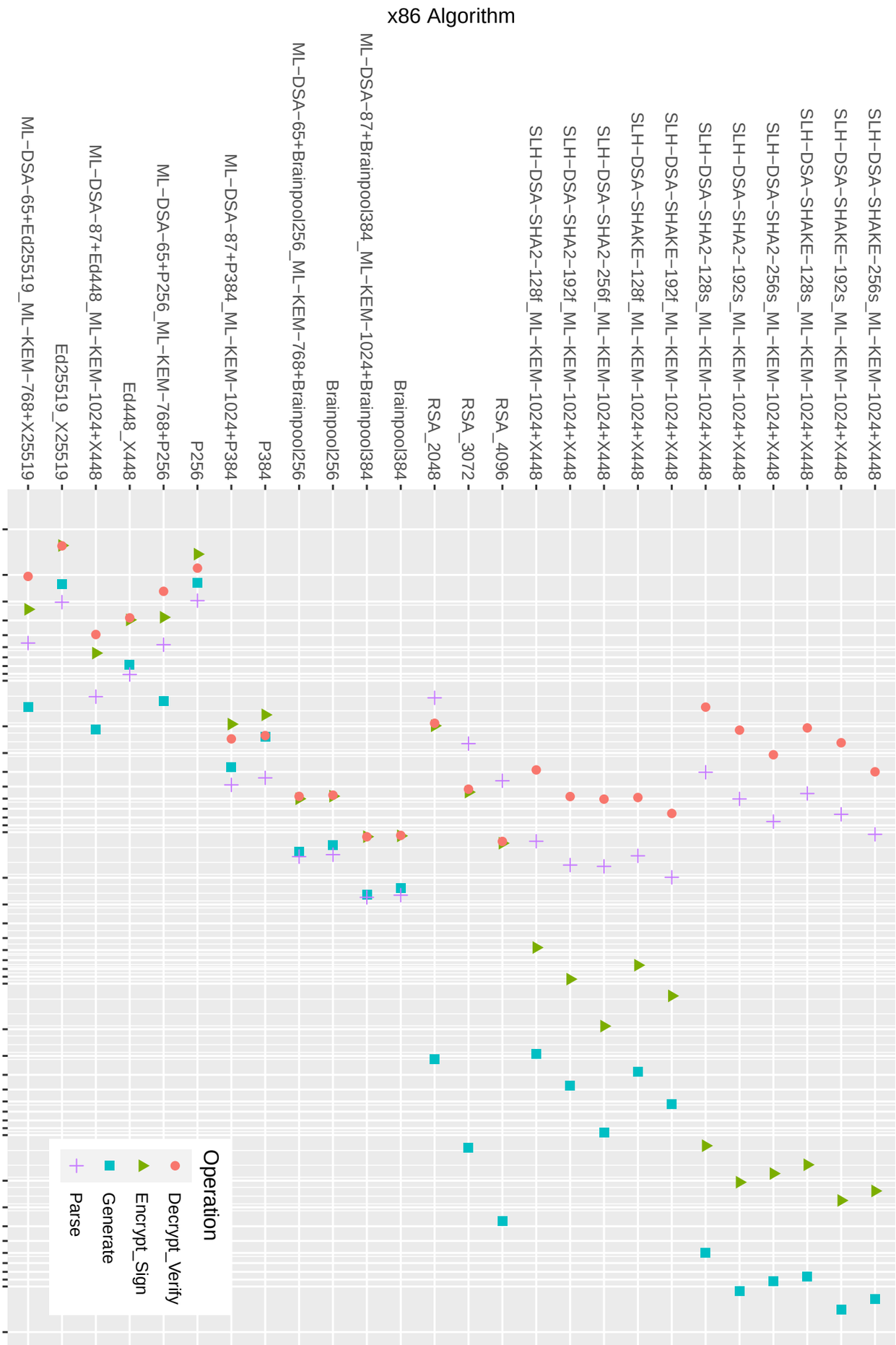

Fig. 2. Overview of the x86 performance for all newly implemented and reference algorithms for 1 KB of data using a round 3 submission implementation. The plot is logarithmic in the operation time.

Fig. 3. Overview of the ARM performance for all newly implemented and reference algorithms for 1 KB of data using a round 3 submission implementation. The plot is logarithmic in the operation time.

11 Acknowledgements

This project was developed by a cooperation of MTG AG, Proton AG, and the German Federal Office for Information Security (BSI). The following people contributed to the project: Falko Strenzke, Stavros Kousidis, Johannes Roth, Andreas Hülsing, Stephan Ehlen, Carl-Daniel Hailfinger, and Evangelos Karatsiolis.

References

[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.

[BCD20] Elaine Barker, Lily Chen, and Richard Davis. Recommendation for key-derivation methods in key-establishment schemes. Technical report, August 2020.

[Cou22] NIST General Counsel. Nist pqc license summary and excerpts, 2022.

[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.

[ELPa] Karen Easterbrook, Brian LaMacchia, and Christian Paquin. Post-quantum tls.

[ELPb] Karen Easterbrook, Brian LaMacchia, and Christian Paquin. Post-quantum tls.

[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.

[GHP18] Federico Giacon, Felix Heuer, and Bertram Poettering. Kem combiners. Cryptology ePrint Archive, Paper 2018/024, 2018.

[KSL+19] Krzysztof Kwiatkowski, Nick Sullivan, Adam Langley, Dave Levin, and Alan Mislove. Measuring tls key exchange with post-quantum kem, 2019.

[KSW23] Stavros Kousidis, Falko Strenzke, and Aron Wussler. Post-Quantum Cryptography in OpenPGP. Internet-Draft draft-wussler-openpgp-pqc-03, Internet Engineering Task Force, October 2023. Work in Progress.

[KV] Krzysztof Kwiatkowski and Luke Valenta. The tls post-quantum experiment.

[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.

[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.

[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.

[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.

[WHWY24] Paul Wouters, Daniel Huigens, Justus Winter, and Niibe Yutaka. OpenPGP. Internet-Draft draft-ietf-openpgp-crypto-refresh-13, Internet Engineering Task Force, January 2024. Work in Progress.