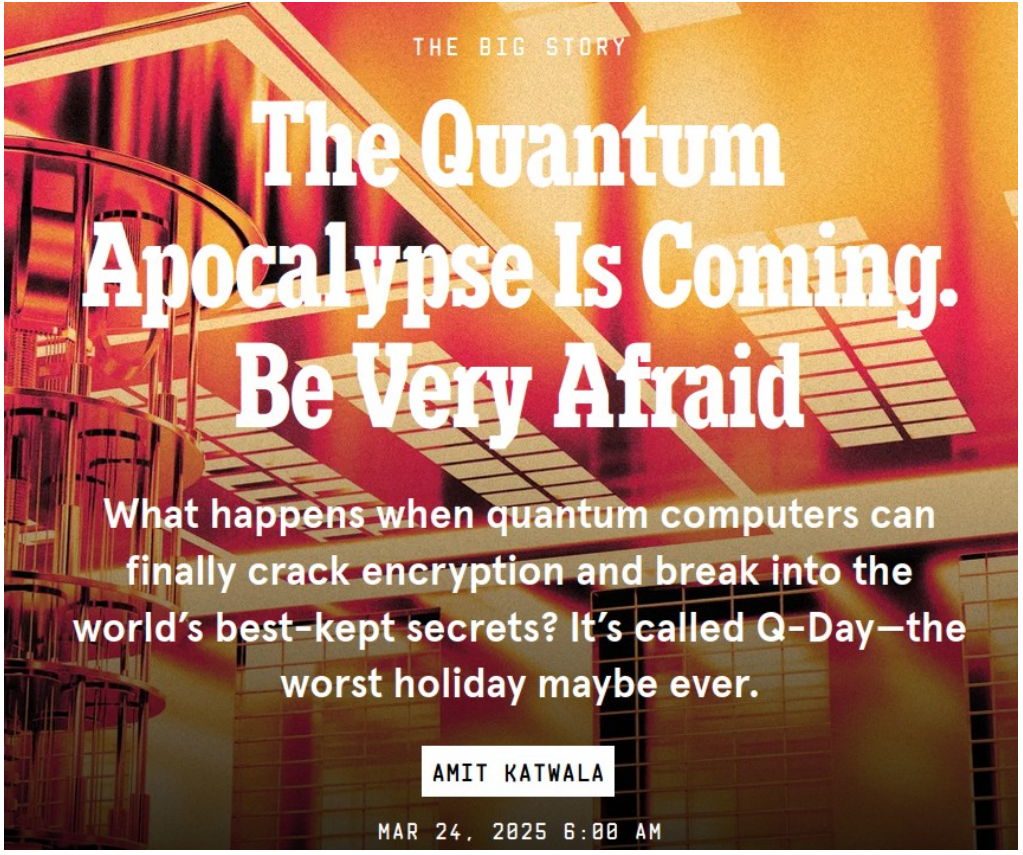
Über Post-Quanten-Kryptografie und den aktuellen Stand von Open-Source-Implementierungen

Eine kleine Einführung

Hans Löhr
Kryptografie und Informationssicherheit
Fakultät Informatik
Technische Hochschule Nürnberg Georg Simon Ohm

Quantenapokalypse?

https://www.wired.com/story/q-day-apocalypse-quantum-computers-encryption/



Quantenapokalypse?

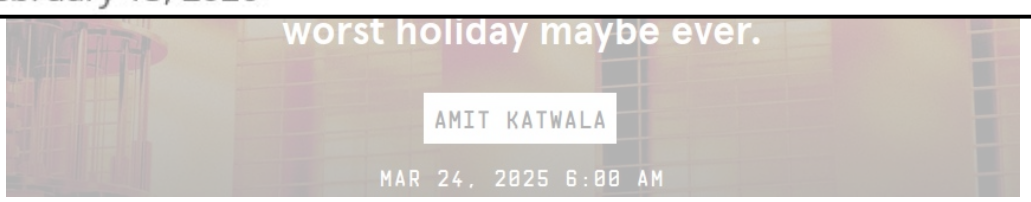
https://www.wired.com/story/q-day-apocalypse-quantum-computers-encryption/

https://www.linkedin.com/pulse/stop-fearing-quantum-apocalypse-start-learning-linear-muhammad-adnan-v7gof/

Stop Fearing the Quantum Apocalypse. Start Learning Linear Algebra

Muhammad Adnan
Founder at QuantumProgramming.tech | Web Development, AI/ML, Cryptography & Technical...

February 15, 2026



Übersicht

- Was ist Post-Quanten-Kryptografie (PQC)?
  - Kurze Begriffsklärung und Abgrenzung
- Quantencomputer und Kryptografie – Wo ist das Problem?
  - Quantencomputer – real oder Science Fiction?
  - Klassische Kryptografie und Quantencomputer
- Post-Quanten-Kryptografie
  - Grundlegende Ansätze
  - NIST-Wettbewerb, Standardisierung
- PQC-Implementierungen
  - PQC im Praxiseinsatz
  - Ausgewählte Crypto-Libraries (open-source), Performance
  - PQC in Rust für eingebettete Systeme
- Fazit, Ausblick

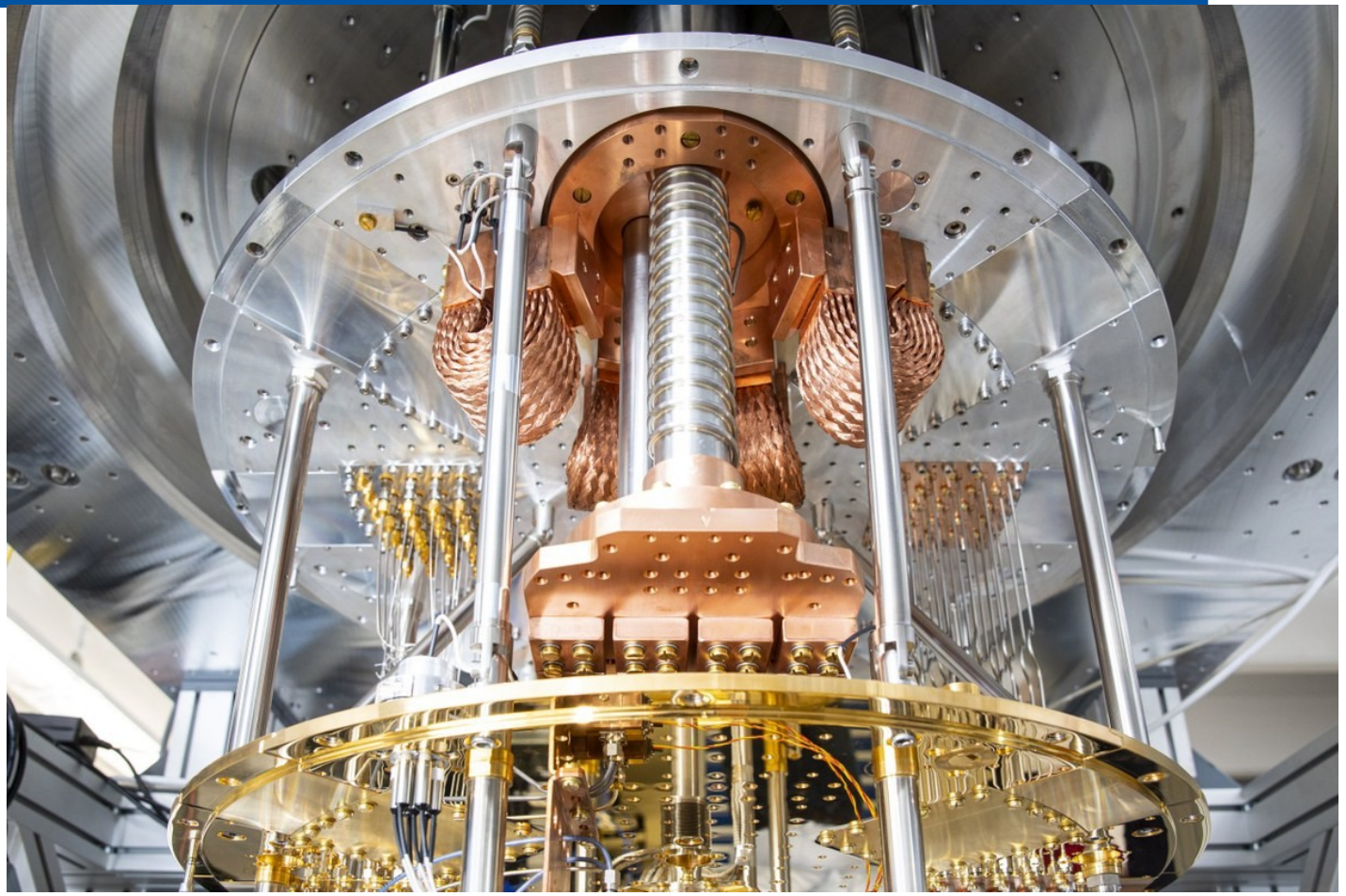
Post-Quanten-Kryptografie
Begriffsklärung

- **Post-Quanten-Kryptografie / quantensichere Kryptografie**
  (engl.: post-quantum cryptography / quantum-safe cryptography)
  - Kryptografie mit klassischen Computern, die auch Angriffen mit Quantencomputern standhalten soll

- Quantenkryptografie (quantum cryptography)
  - Kryptografie unter Verwendung von Quantentechnologie
  - insbesondere Quantum Key Distribution (QKD)

Quantencomputer

[Bildquelle: Forschungszentrum Jülich / Sascha Kreklau, 2023]



Quantencomputer (QC)

- QC rechnen mit qbits statt bits
  - Nicht nur 1 und 0
  - Superposition und Verschränkung möglich
  - In etwa: gleichzeitig mehrere Zustände
- Schaltkreise aus Quantengattern statt klassischen logischen Gattern
- Ergebnis einer Berechnung: Messung
- Verschiedene physikalische Realisierungen möglich
- Aktuell: bis zu hundert(e) qbits realisierbar
  - Problem: Rauschen, Fehlerkorrektur
    => noch nicht ausreichend für Kryptoanalyse

Quantenalgorithmen: Grover, Shor

- **Grovers Algorithmus / Grover Search** (Lov Grover, 1996):
  - Generischer Suchalgorithmus
  - Komplexität: O($\sqrt{N}$)
    (N: Größe der Input Domain)
  - Kryptografische Anwendungen: *brute-force*-Angriffe beschleunigen
    - Quadratischer speed-up gegenüber klassischen Algorithmen
    - Bsp.: *n* bit durchprobieren mit Grover Search
      entspricht
      n/2 bit durchprobieren klassisch
- **Shors Algorithmus** (Peter Shor, 1994):
  - Faktorisierungsalgorithmus: Zahl in Primfaktoren zerlegen
  - Polynomielle Laufzeit
  - Varianten zur Berechnung des diskreten Logarithmus’

Klassische Kryptografie
Symmetrische Algorithmen

- Symmetrische Verschlüsselung (Blockchiffren, Stromchiffren)
  - Bsp. AES *(Advanced Encryption Standard)*
- Kryptografische Hashfunktionen
  - Bsp. SHA-256, SHA-3 *(Secure Hash Algorithm)*
- Message Authentication Codes
  - Bsp. CMAC, HMAC

- ***Angreifbar mit Grover Search!***
- Grobe Faustregel: **Sicherheitsparameter** (z.B. Schlüssellänge) **verdoppeln => sicher** gegen Angriffe mit Quantencomputern

Das RSA-Kryptosystem („Textbook RSA")

- Schlüsselgenerierung:
  - Wähle zufällige, große (unterschiedliche) Primzahlen p, q (etwa gleicher Größe)
  - Berechne $n = p \cdot q$ und $\varphi(n) = (p-1) \cdot (q-1)$
  - Wähle zufällige, große Zahl $e \in \mathbb{Z}$ mit $1 < e < \varphi(n)$, so dass $ggT(e, \varphi(n)) = 1$
  - Berechne d mit $1 < d < \varphi(n)$, so dass $e \cdot d \equiv 1 \pmod{\varphi(n)}$
  - *Public key* $PK := (n, e)$, *private key* $SK := d$
- Verschlüsselung: $c := \mathtt{enc}_{PK}(m) := m^e \pmod{n}$
- Entschlüsselung: $m' := \mathtt{dec}_{SK}(c) := c^d \pmod{n}$

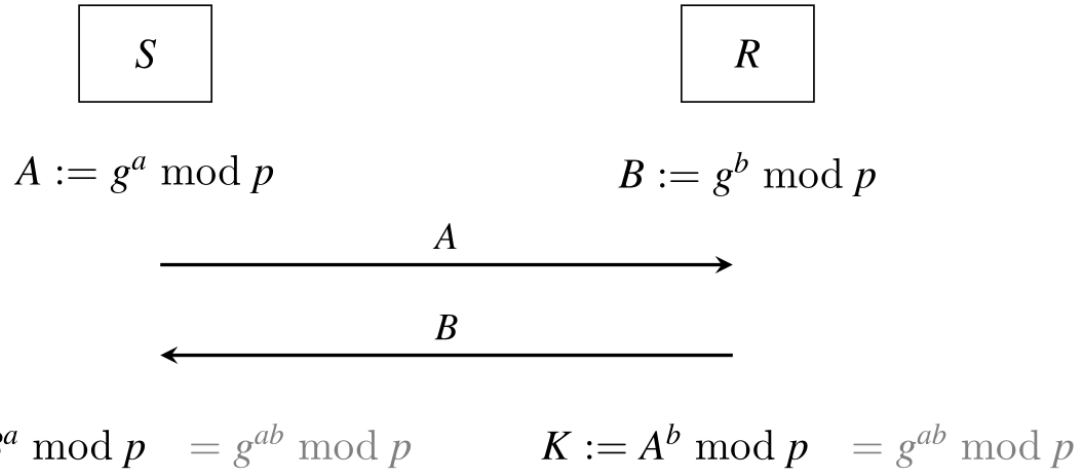
Klassische Kryptografie
Asymmetrisch: RSA-Verschlüsselung

Das RSA-Kryptosystem („Textbook RSA")

- Schlüsselgenerierung:
  - Wähle zufällige, große (unterschiedliche) Primzahlen p, q (etwa gleicher Größe)
  - Berechne $n = p \cdot q$ und $\varphi(n) = (p-1) \cdot (q-1)$
  - Wähle zufällige, große Zahl $e \in \mathbb{Z}$ mit $1 < e < \varphi(n)$, so dass $ggT(e, \varphi(n)) = 1$
  - Berechne d mit $1 < d < \varphi(n)$, so dass $e \cdot d \equiv 1 \pmod{\varphi(n)}$
  - *Public key* $PK := (n, e)$, *private key* $SK := d$
- Verschlüsselung: $c := \mathtt{enc}_{PK}(m) := m^e \pmod{n}$
- Entschlüsselung: $m' := \mathtt{dec}_{SK}(c) := c^d \pmod{n}$

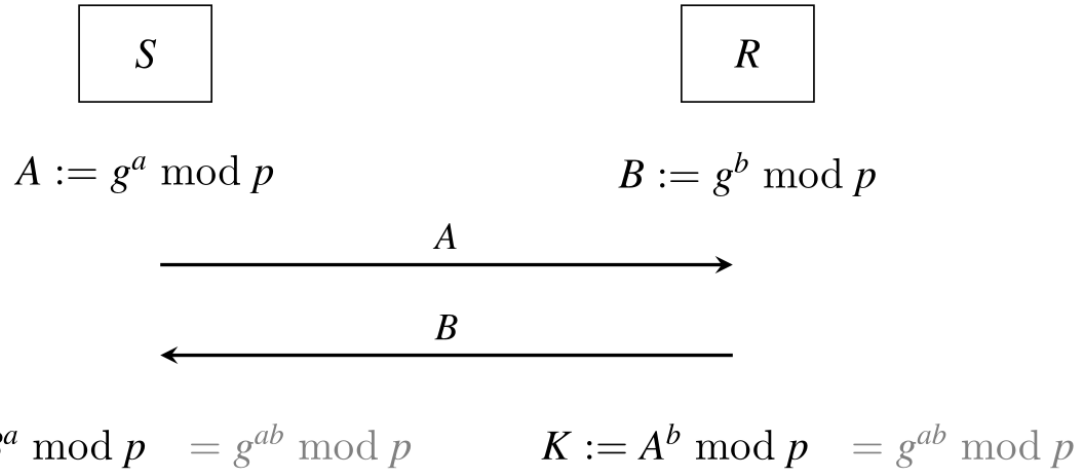
- Faktorisierung von n (mit Shor) => privaten Schlüssel berechnen (gilt ebenso für RSA-Signaturen)

Klassische Kryptografie
Asymmetrisch: Diffie-Hellman Key Agreement



- Berechnungen in zyklischer Gruppe G mit Generator g
  Hier: modulo Primzahl p, in von g erzeugter Untergruppe von $\mathbb{Z}_p^*$: $G = \langle g \rangle \subseteq \mathbb{Z}_p^*$
- g, p sind öffentliche Parameter
- a, b werden zufällig gewählt und bleiben geheim (werden nicht übertragen)
- Schlüsselvereinbarung: beide Parteien berechnen gleichen Schlüssel K
- Sicherheit beruht auf DH-Annahme *(DH assumption)*
  Intuitiv: g^{ab} ist „schwierig zu berechnen" (bei Kenntnis von g^a, g^b)
  oder „schwierig von g^r (r zufällig) zu unterscheiden"

Klassische Kryptografie
Asymmetrisch: Diffie-Hellman Key Agreement



- Berechnungen in zyklischer Gruppe G mit Generator g
  Hier: modulo Primzahl p, in von g erzeugter Untergruppe von $\mathbb{Z}_p^*$: $G = \langle g \rangle \subseteq \mathbb{Z}_p^*$
- g, p sind öffentliche Parameter
- a, b werden zufällig gewählt und bleiben geheim (werden nicht übertragen)
- Schlüsselvereinbarung: beide Parteien berechnen gleichen Schlüssel K
- Sicherheit beruht auf DH-Annahme *(DH assumption)*
  Intuitiv: g^{ab} ist „schwierig zu berechnen" (bei Kenntnis von g^a, g^b)
  oder „schwierig von g^r (r zufällig) zu unterscheiden"

Diskreter Logarithmus gebrochen
=> DH-Annahme gebrochen

Klassische Kryptografie
Weitere asymmetrische Algorithmen

- ElGamal, DSA
  - Sicherheit beruht auch auf diskretem Logarithmus (modulo Primzahl)
    => auch durch Shors Algorithmus brechbar!
- Algorithmen auf elliptische Kurven
  - Sicherheit beruht auf diskretem Logarithmus in anderen Gruppen ...
    => auch durch Shors Algorithmus brechbar!

- **Fazit:** *Shors Algorithmus bedroht **alle** weitverbreiteten klassischen asymmetrischen Kryptosysteme!*

Post-Quanten-Kryptografie – Jetzt schon?

- OK, ***wenn*** (geeignete, leistungsstarke) ***Quantencomputer*** gebaut werden können, ***dann*** brauchen wir ***“neue”*** (asymmetrische) ***Kryptografie***.

  ... aber ab wann? ... jetzt schon??

Post-Quanten-Kryptografie – Jetzt schon?

- OK, ***wenn*** (geeignete, leistungsstarke) ***Quantencomputer*** gebaut werden können, ***dann*** brauchen wir ***“neue”*** (asymmetrische) ***Kryptografie***.

  ... aber ab wann? ... jetzt schon??

- ***Niemand*** kann seriös sagen, ***wann*** *(evtl. sogar ob?)* es soweit sein wird.

Post-Quanten-Kryptografie – Jetzt schon?

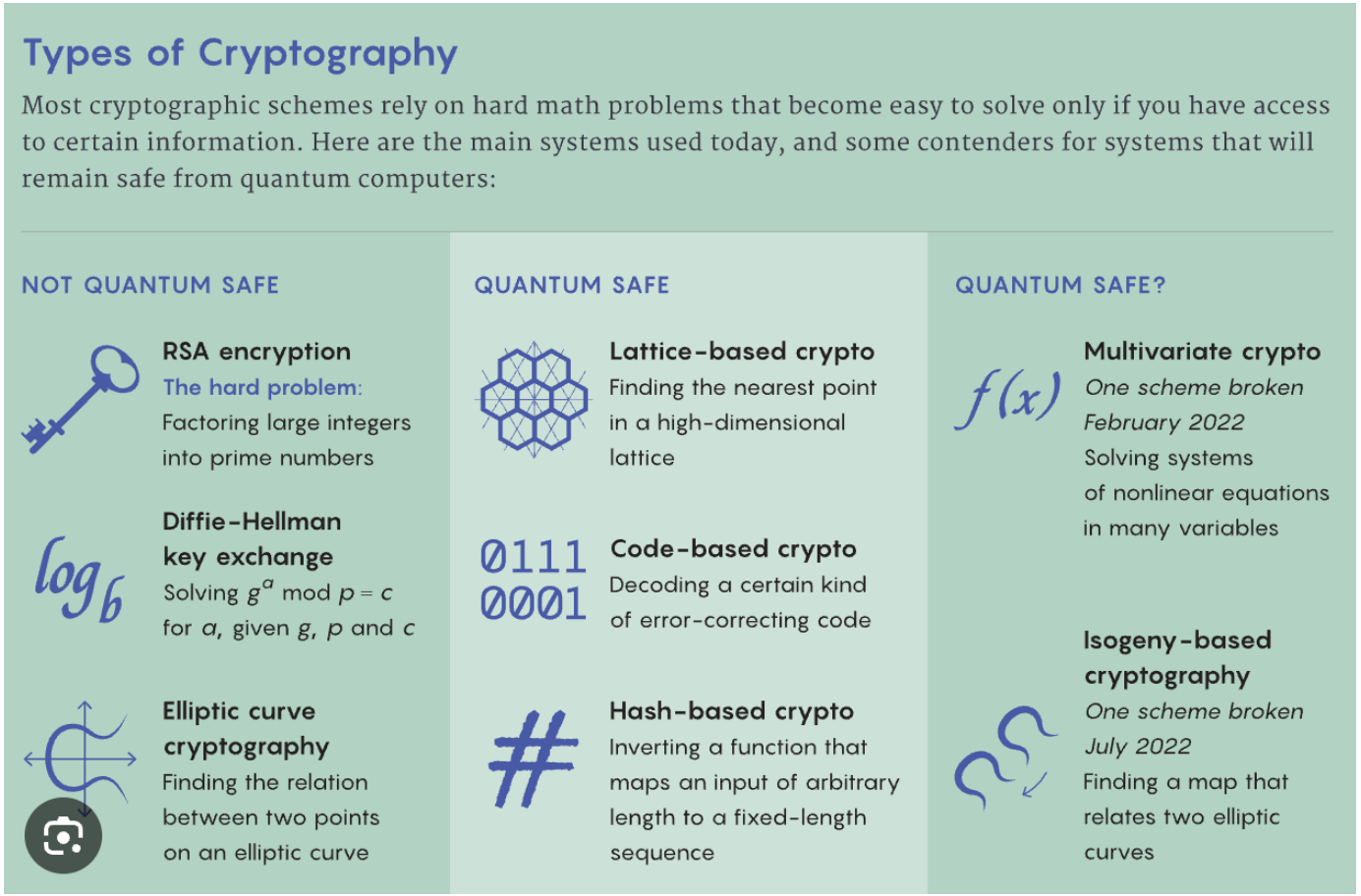
- OK, ***wenn*** (geeignete, leistungsstarke) ***Quantencomputer*** gebaut werden können, ***dann*** brauchen wir ***“neue”*** (asymmetrische) ***Kryptografie***.

  ... aber ab wann? ... jetzt schon??

- ***Niemand*** kann seriös sagen, ***wann*** *(evtl. sogar ob?)* es soweit sein wird.
- Aber:

  “Store now, decrypt later!”

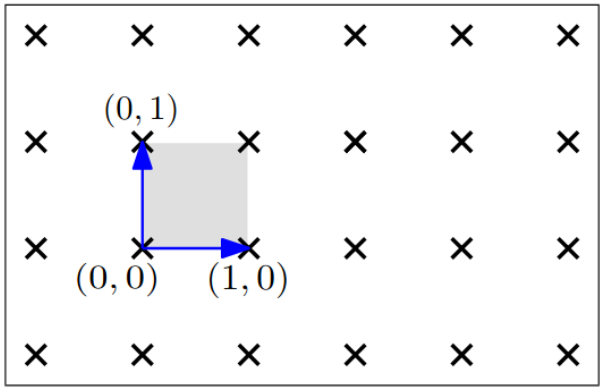
Post-Quanten-Kryptografie: Harte Probleme für Quantencomputer?



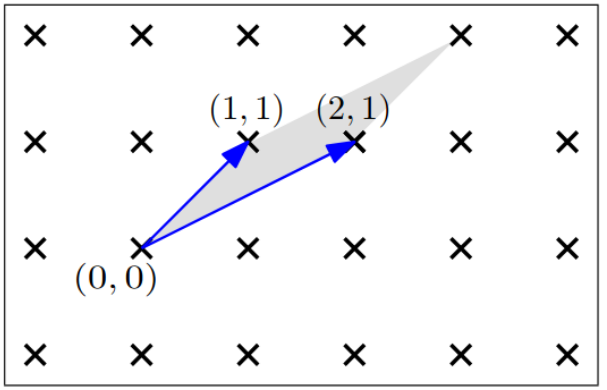
[Bildquelle: Quanta Magazine, 2023]

PQC: Gitterbasierte Kryptografie
Lattice-based Cryptography

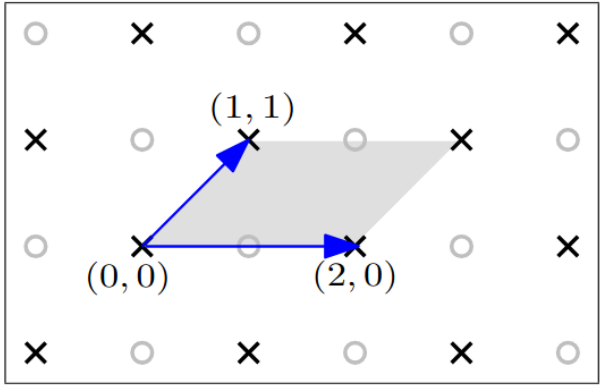
- Gitter *(Lattice)*
  - Diskrete Struktur
  - Von Basisvektoren aufgespannt (ganzzahlige Linearkombinationen)
  - Basis nicht eindeutig
  - Verallgemeinert auf Dimensionen > 2



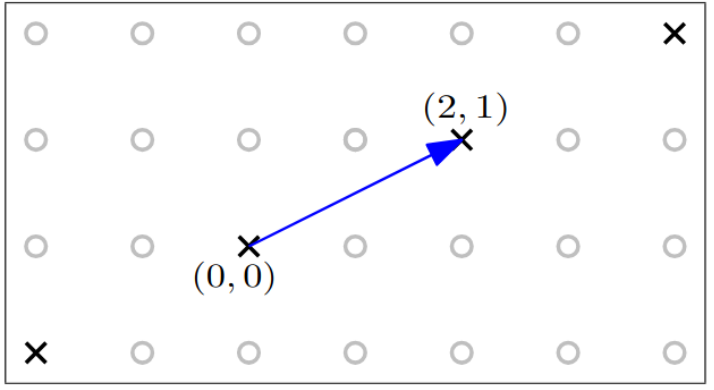
(a) A basis of $\mathbb{Z}^2$



(b) Another basis of $\mathbb{Z}^2$



(c) Not a basis of $\mathbb{Z}^2$



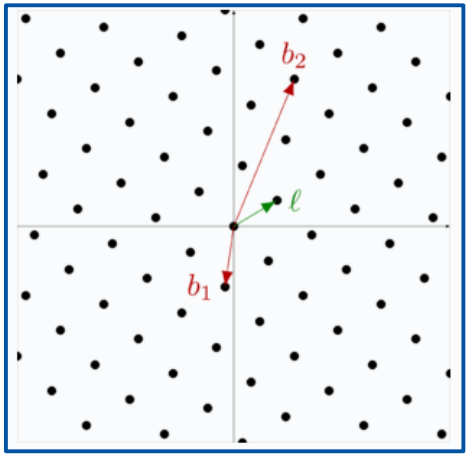
(d) Not a full-rank lattice

[Bildquelle: D. Dadush, L. Ducas: Lecture "Introduction to Lattice Algorithms and Cryptography", CWI, Amsterdam, NL, 2018]

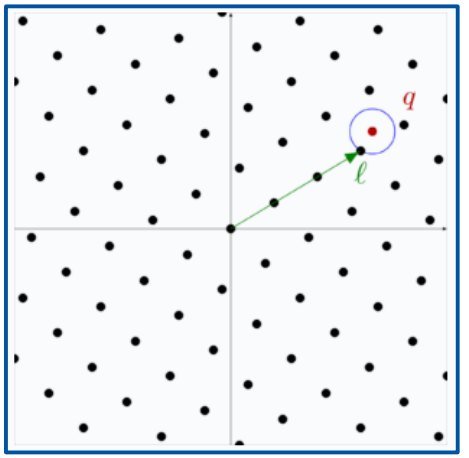
PQC: Gitterbasierte Kryptografie (vermutlich) harte Probleme

[Bildquelle: Richter, Seidensticker, Bertram: “A (somewhat) gentle Introduction to lattice-based PQC”, Fraunhofer AISEC, 2023]

- *Shortest Vector Problem (**SVP**):*
  Finde kürzesten Vektor in einem Gitter (gegeben durch eine Basis)
- *Closest Vector Problem (**CVP**):*
  Finde nächstliegenden Gitterpunkt zu gegebenem reellem Vektor (in durch Basis gegebenem Gitter)



SVP



CVP

- **Annahme:** *SVP* und *CVP* sind harte Berechnungsprobleme – auch für Quantencomputer (Shors Algorithmus ist nicht anwendbar)

PQC: Gitterbasierte Kryptografie
(vermutlich) harte Probleme

- *Effizient berechenbar:* Gegeben Matrix *A*, Vektor *b* (ganzzahlig, modulo *q*), finde Lösung s (ganzzahlig, modulo q), so dass gilt:

$$A \cdot s = b$$

- *Learning with Errors (**LWE**) Problem:*
  Finde ganzzahlige Lösung ***(s,e)*** modulo ***q*** der Gleichung (modulo *q*)

$$A \cdot s + e = b$$

  bei gegebener Matrix *A*, Vektor *b* (alles ganzzahlig, modulo *q*)

- **LWE-Annahme:**
  LWE ist hartes Berechnungsprobleme – auch für Quantencomputer
  --> beweisbar, falls (gewisse) andere Gitterprobleme hart sind

PQC: (vermutlich) harte Probleme für realistische Kryptosysteme

- *Learning with Errors (LWE)* kann als Basis für Kryptosysteme verwendet werden
  - Bsp.: **Frodo** (aber: relativ ineffizient)

- Varianten von *LWE* existieren (mathematisch etwas komplizierter)
  - *Ring-LWE*, *Module-LWE*
  - Für moderne, effiziente Kryptosysteme verwendet
    - ***Module-LWE*** gilt als (wahrscheinlich) sicherer als *Ring-LWE* (bei gleicher Parametrisierung)
    - Bsp.: **Kyber**

- Neben gitterbasierten Problemen gibt es auch andere Kandidaten, insbesondere
  - aus der *Codierungstheorie* (Bsp.: **McEliece** mit Varianten)
  - basierend auf *kryptographischen Hashfunktionen* (Bspe: **SPHINCS+**, **XMSS**)

NIST-Wettbewerb, Standardisierung

- Bis ca. 2015: PQC eher noch Nischenthema
- 2016: NIST startet Standardisierungsprozess: offener Wettbewerb
- Juli 2022: erste Standardisierungskandidaten (nach 3 Runden Wettbewerb):
  - **Crystals-Kyber** (key encapsulation mechanism / Verschlüsselung) -> **ML-KEM** (FIPS 203)
    (gitterbasiert, Module-LWE)
  - **Crystals-Dilithium** (digitale Signatur – generisch) -> **ML-DSA** (FIPS 204)
    (gitterbasiert, Module-LWE)
  - **FALCON** (digitale Signatur – kurze Signatur) -> **FN-DSA** (FIPS 206, noch Draft)
    (gitterbasiert, FFT NTRU)
  - **SPHINCS+** (digitale Signatur – alternativ) -> **SLH-DSA** (FIPS 205)
    (stateless Hash-based)
- August 2024: erste Standards finalisiert (FIPS 203, 204, 205)
- März 2025: **HQC** als weiterer **KEM** ausgewählt (basierend auf *error-correcting codes*)
- Zweiter Call für weitere Signatur-Algorithmen läuft seit August 2022:
  aktuell Runde 2 (14 Kandidaten)

BSI-Empfehlungen zur PQC-Migration
Bundesamt für Sicherheit in der Informationstechnik

- BSI empfiehlt **Migration** sensitiver Anwendungen auf quantenresistente Verfahren **bis 2030**.
- BSI setzt aktuell auf **hybride** Lösungen:
  - Kombination aus klassischer Crypto und PQC
  - ... so, dass Angreifer *beide* Kryptosysteme brechen muss, um erfolgreich zu sein

- => ... wie ist der aktuelle Stand in der Praxis?
  - realer Einsatz von PQC?
  - Unterstützung in Software-Libraries?

PQC: Praktischer Einsatz hat begonnen
Beispiele

- TLS: PQC (hybride Lösung) schon lange erprobt
  - z.B. Google, Cloudflare, Mozilla, andere
  - Für Webbrowsing: kein merklicher Performance-Nachteil
- Instant messaging (z.B. Signal, Apple)
  - hybride Lösung
  - Endnutzer merken nichts davon
- OpenSSH: hybrider key exchange
- OpenPGP: IETF Draft, Implementierungen

Stand PQC-Unterstützung in Crypto Libraries? Untersuchung durch studentisches Projekt 2025

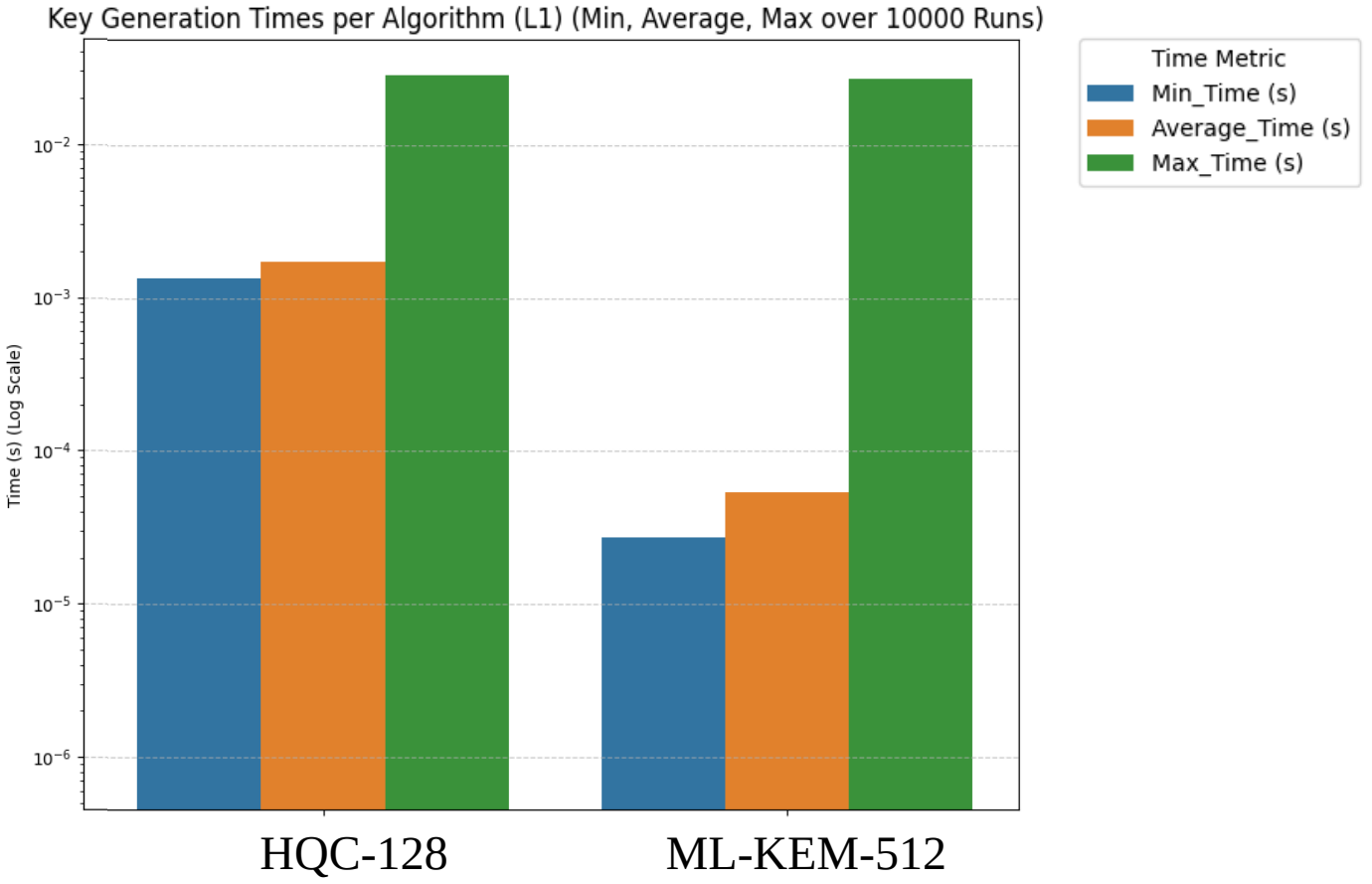
- Projektteam: Sher Omar Abdi, Jason Bell, Alexander Bounnhong, Kristina Cutler, Stefan Tischkin
- Zielsetzung: Praxistauglichkeit testen, Performance-Messungen

- Untersuchte Bibliotheken
  - **liboqs** (Open Quantum Safe Project, s. openquantumsafe.org)
  - **botan** (verbreitete C++ Crypto-Library)

- Testumgebung für Performance-Messungen
  - VirtualBox 7.1.4, Ubuntu 25.04
  - 6000 MB RAM, 2 CPUs, 50 GB hard drive
  - Python und Jupyter Notebook

PQC in Crypto Libraries: liboqs
KEM

- Key Encapsulation Mechanisms (KEM)
- NIST “Security Category 1”
- Schlüsselgenerierung (Zeit)



PQC in Crypto Libraries: liboqs
Key Encapsulation Mechanisms (KEM)

	PK length (bytes)	SK length (bytes)	KeyGen avg. time (s)	encap. key length (bytes)	encap. avg. time (s)	decap. avg. time (s)
ML-KEM-512	800	1632	0.000054	768	0.000053	0.000037
HQC-128	2249	2305	0.001728	4433	0.003436	0.006139

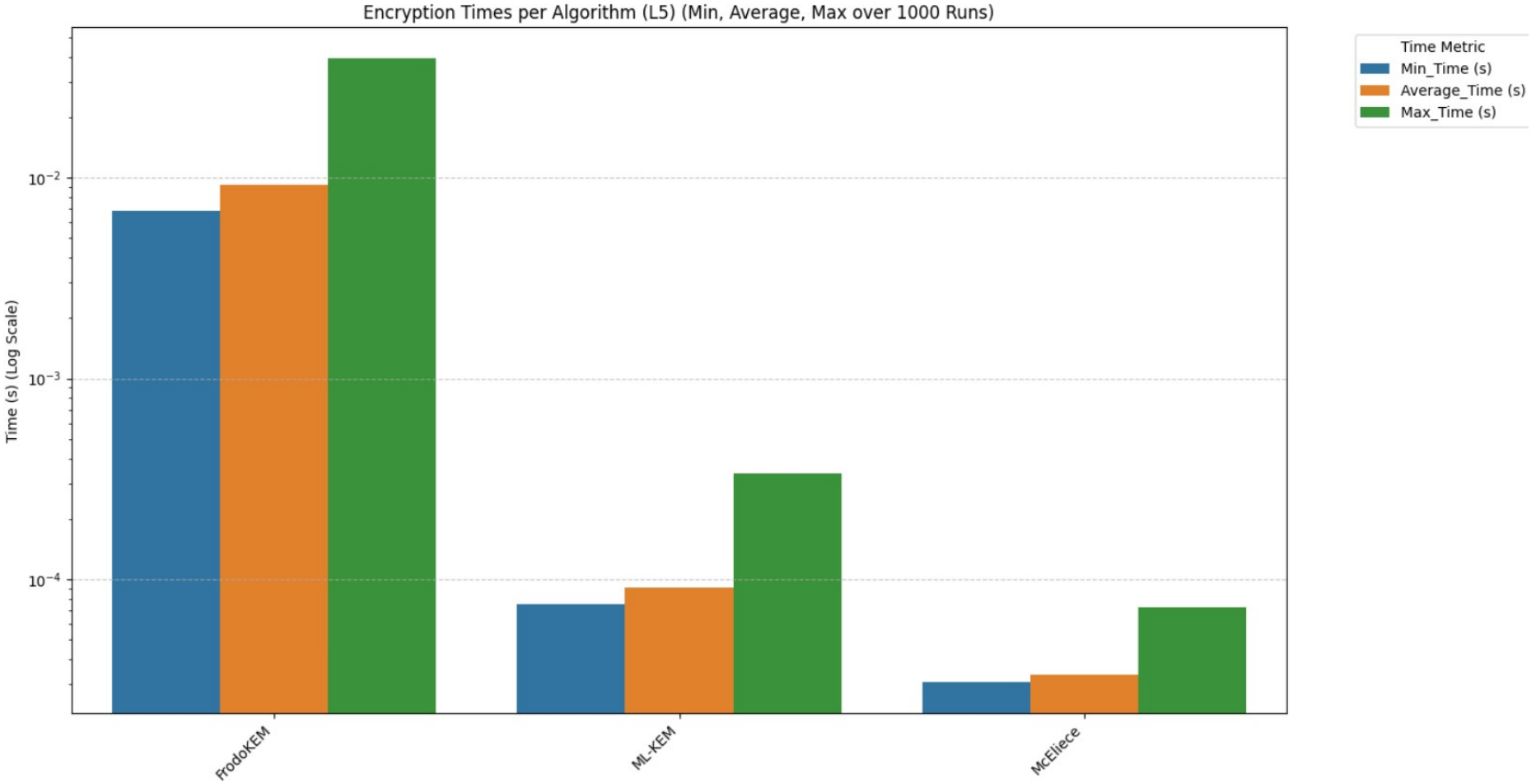
	PK length (bytes)	SK length (bytes)	KeyGen avg. time (s)	encap. key length (bytes)	encap. avg. time (s)	decap. avg. time (s)
ML-KEM-768	1184	2400	0.000065	1088	0.000065	0.000076
HQC-192	4522	4586	0.005794	8978	0.010056	0.016591

	PK length (bytes)	SK length (bytes)	KeyGen avg. time (s)	encap. key length (bytes)	encap. avg. time (s)	decap. avg. time (s)
ML-KEM-1024	1568	3168	0.000070	1568	0.000065	0.000066
HQC-256	7245	7317	0.008733	14421	0.017296	0.029272

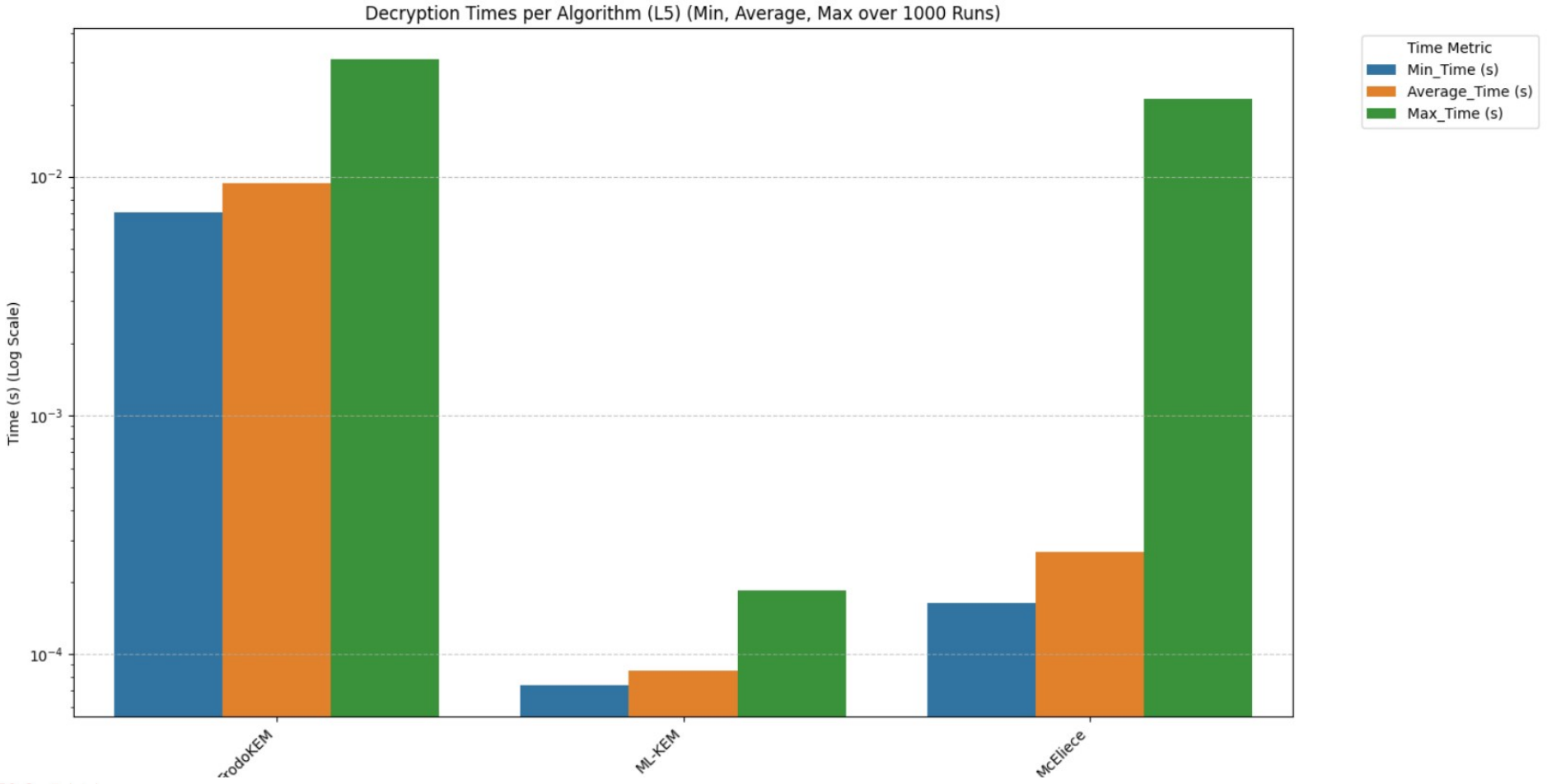
PQC in Crypto Libraries: botan
Key Encapsulation Mechanisms (KEM)

- ML-KEM, FrodoKEM, McEliece
  - Andere Algorithmen, daher schlecht direkt vergleichbar mit liboqs
- FrodoKEM
  - deutlich am langsamsten
- McEliece
  - deutlich größte Schlüssel (z.B. ~200KB Public Keys)
  - aber: am schnellsten bei der key encapsulation

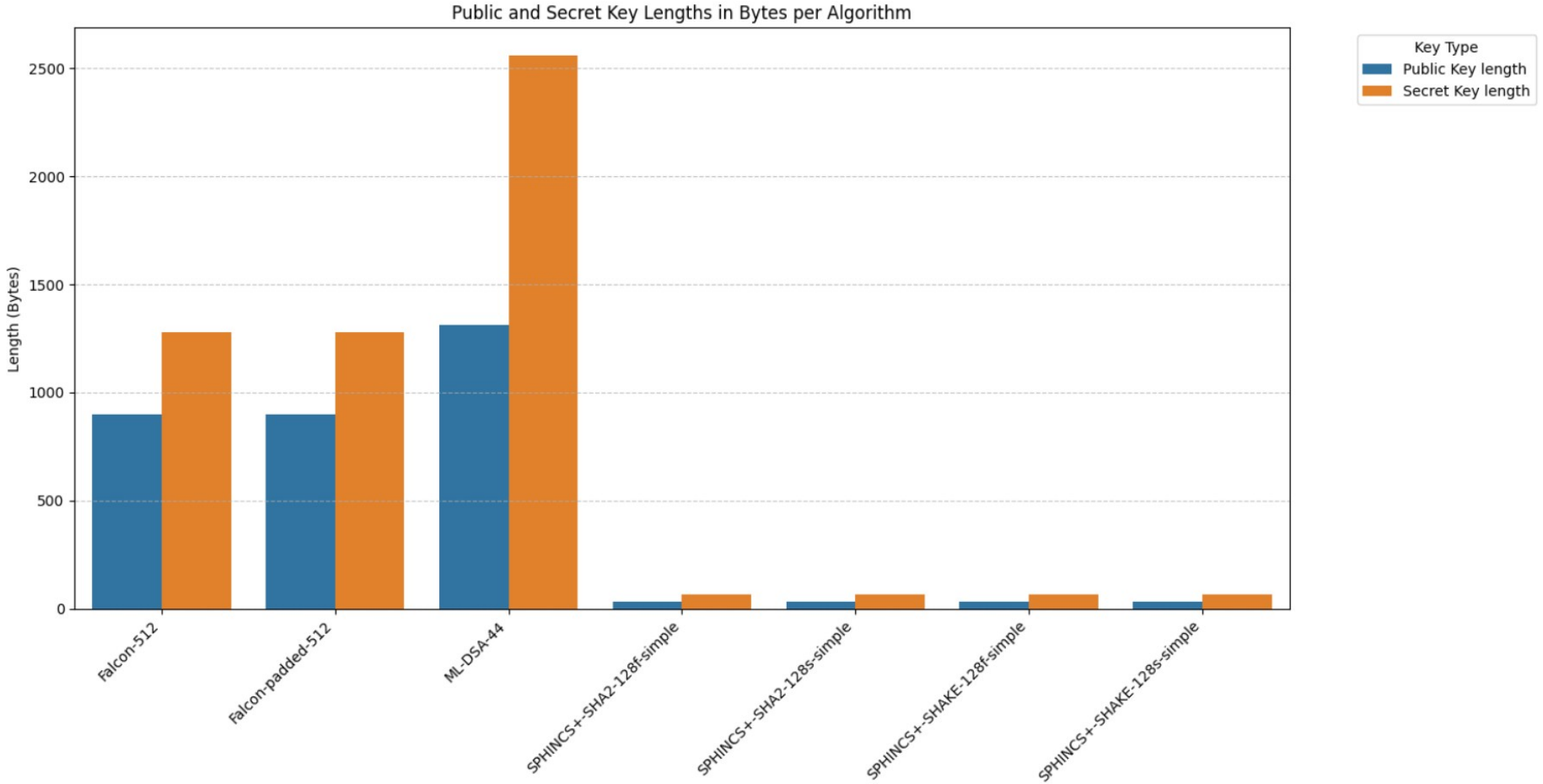
PQC in Crypto Libraries: botan
Key Encapsulation Mechanisms (KEM)



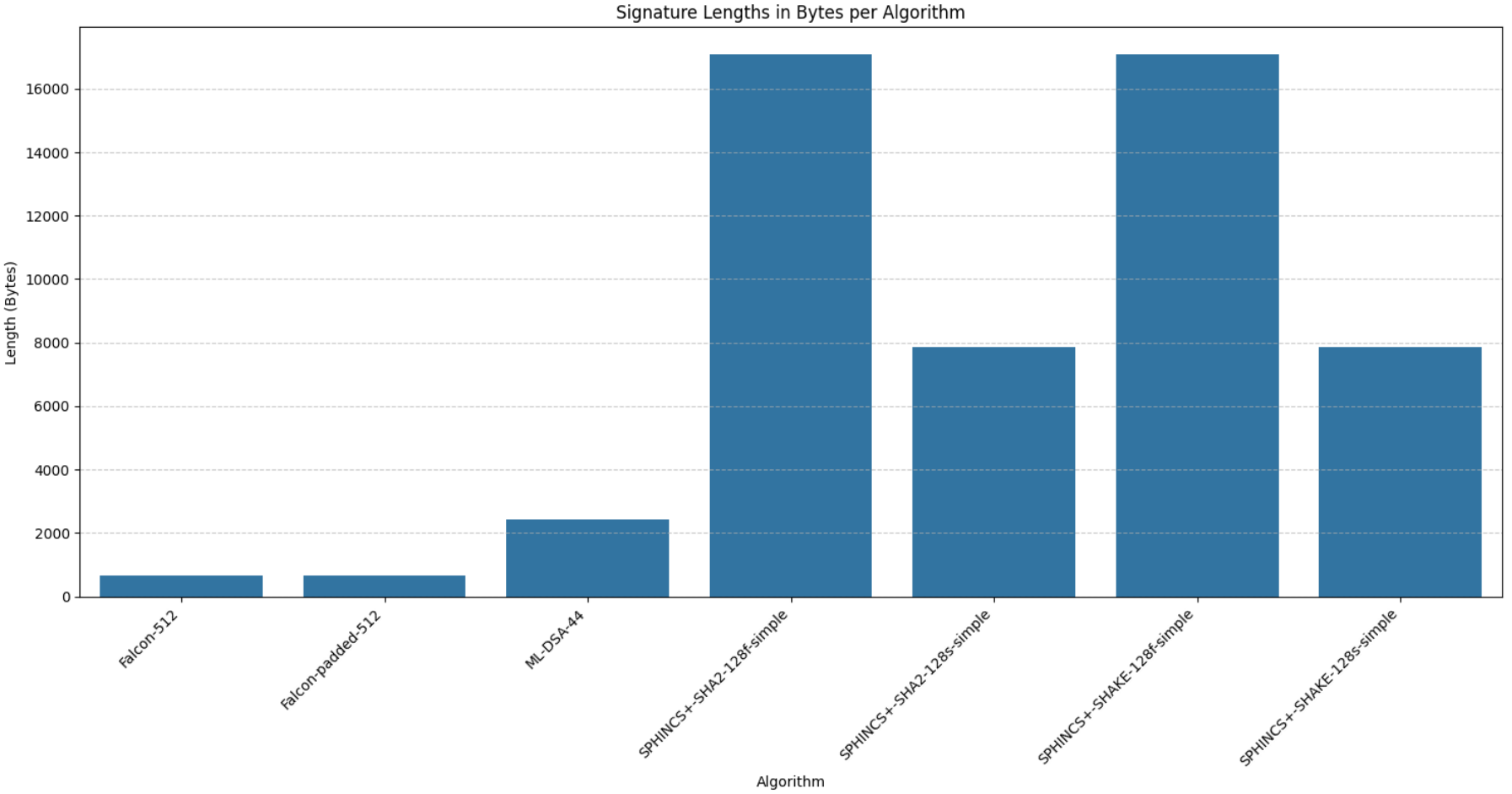
PQC in Crypto Libraries: botan
Key Encapsulation Mechanisms (KEM)



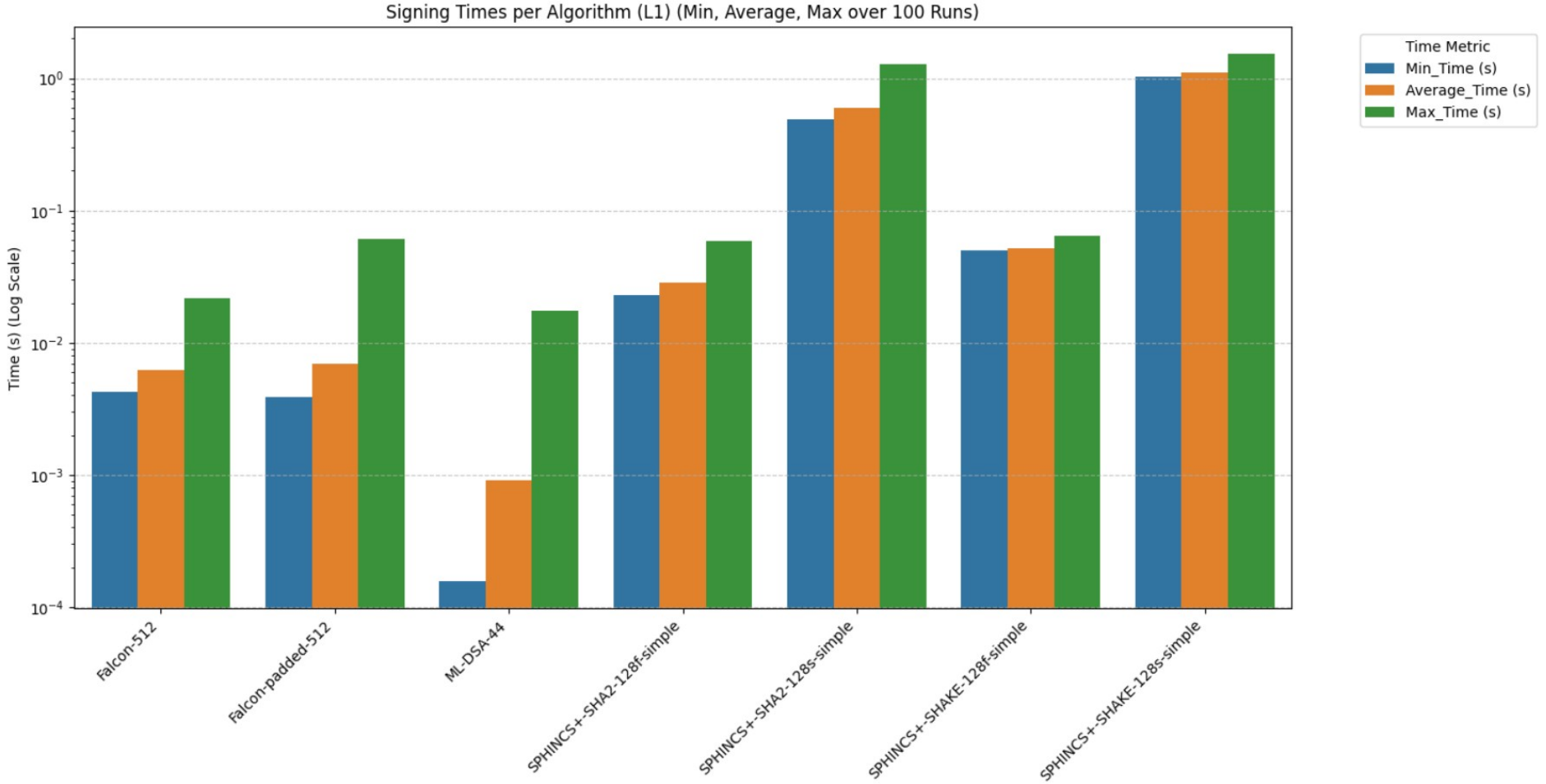
PQC in Crypto Libraries: liboqs
Digitale Signaturen: Schlüssellänge



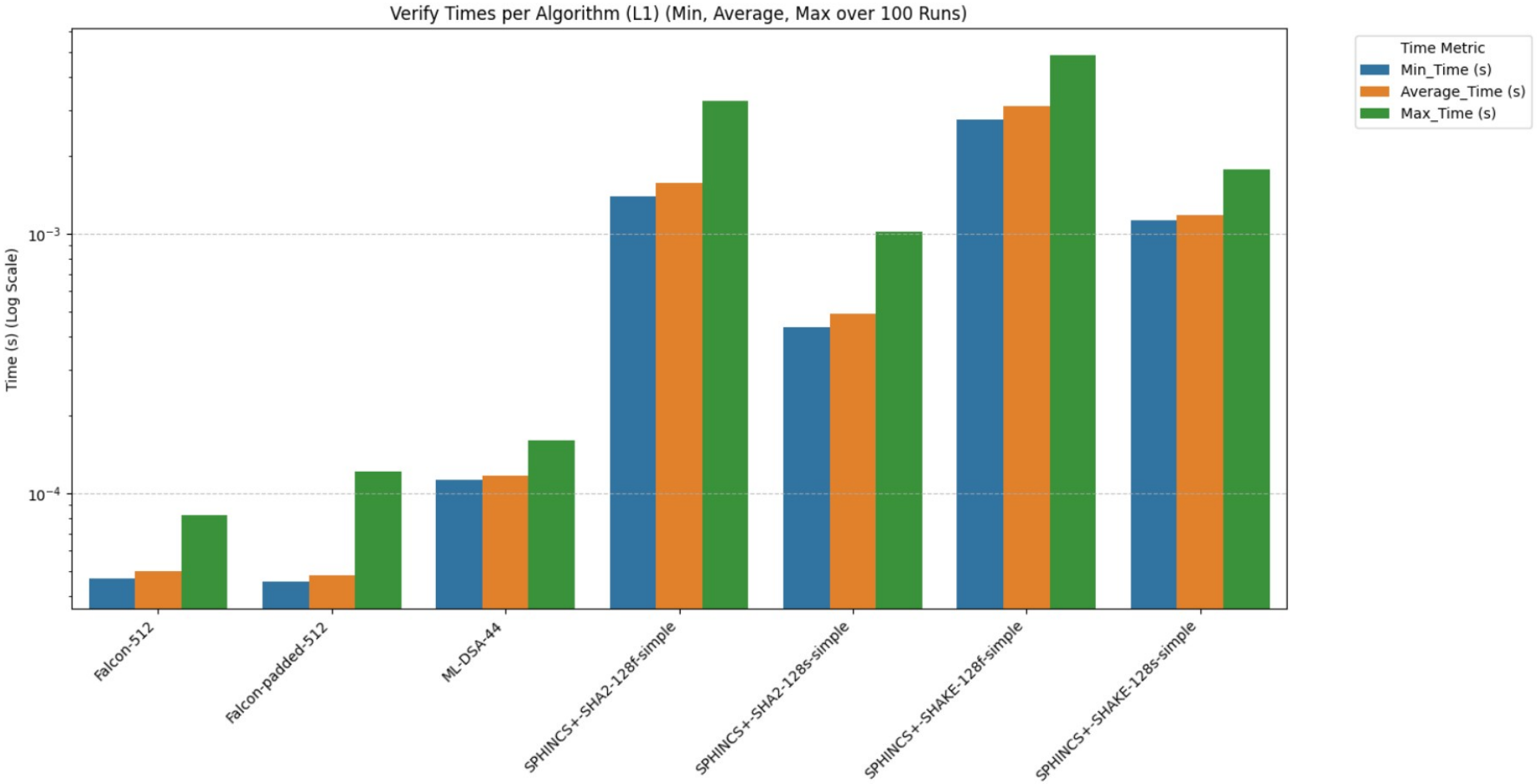
PQC in Crypto Libraries: liboqs
Digitale Signaturen: Signaturlänge



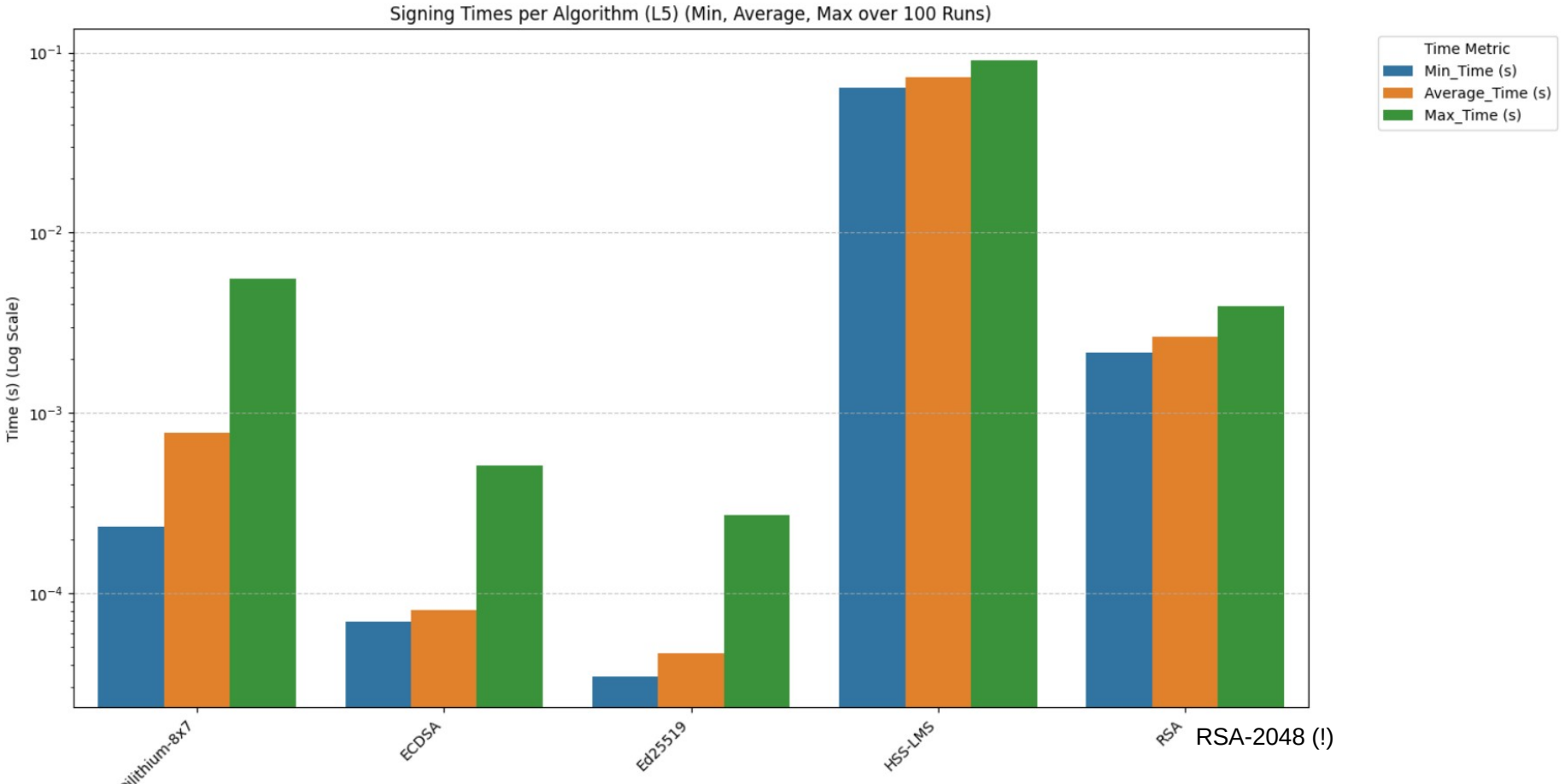
PQC in Crypto Libraries: liboqs
Digitale Signaturen: Signieren (Zeit)



PQC in Crypto Libraries: liboqs
Digitale Signaturen: Verifizieren (Zeit)

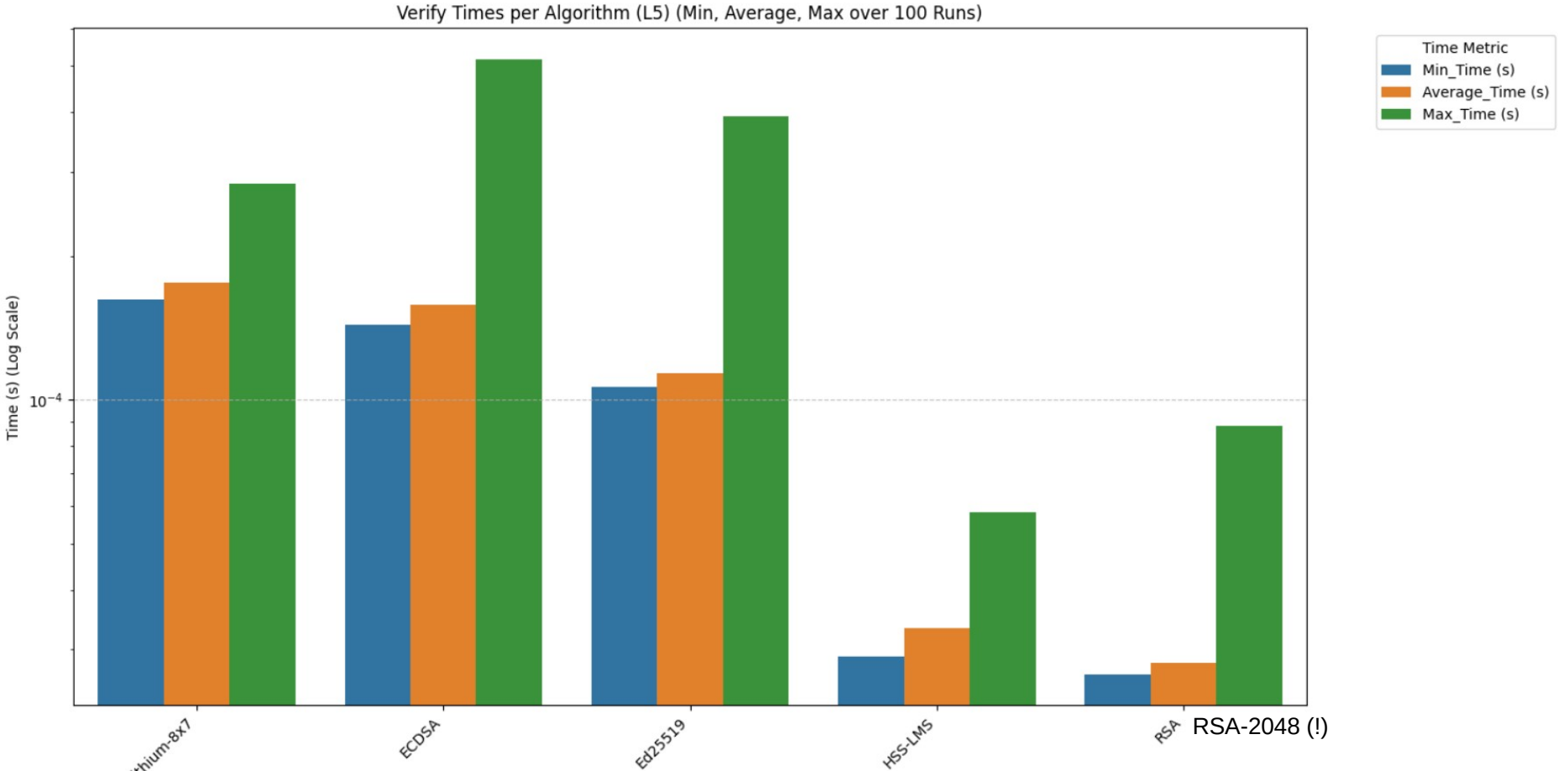


PQC in Crypto Libraries: botan
Digitale Signaturen: Signieren (Zeit)



RSA-2048 (!)

PQC in Crypto Libraries: botan
Digitale Signaturen: Verifizieren (Zeit)



Fazit aus Studierendenprojekt zu PQC

- Studis ohne Erfahrung mit PQC konnten
  - PQC verwenden (sowohl digitale Signaturen als auch KEM)
  - Demoszenarien aufbauen (z.B. PDF-Dokument signieren, verifizieren)
  - Performance-Messungen durchführen
- Die beiden ausgewählten Libraries liboqs, botan
  - unterstützen unterschiedliche Algorithmen (liboqs: keine klassische Crytpo)
  - ließen sich beide relativ problemlos nutzen

Fallstudie: PQC für Rust-Programme auf eingebetteten Systemen (esp-rs)

- Bachelorarbeit: Jonas Lenartz (Arbeit läuft noch)
  - Rust auf kleiner Embedded-Plattform: ESP32-WROOM32D
  - Ziel: Performance-Messungen zu standardisierten Algorithmen ML-KEM, ML-DSA, SLH-DSA
    - liboqs mit Rust-bindings
    - libcrux (rust library)
    - rustpq (rust library)
    - slh-dsa (native rust crate speziell für SLH-DSA)
    - (verworfen, da nicht direkt verwendbar: botan, rust crate ml-dsa)

Fallstudie: PQC für Rust-Programme auf eingebetteten Systemen (esp-rs)

- Bachelorarbeit: Jonas Lenartz
  Vorläufige Ergebnisse
  - ML-KEM
    - rustpq am schnellsten; aber: evtl. nicht zukunftssicher, ML-DSA instabil, hybrider Modus kompiliert nicht...
    - liboqs langsamer, aber scheint solide
  - ML-DSA
    - nur liboqs compiliert, läuft und verarbeitet Testvektoren (ACVP) korrekt
  - SLH-DSA
    - liboqs am schnellsten
    - rust crate slh-dsa langsamer, aber geringere zeitliche Abweichungen

Fazit

- Umstieg auf (hybride) Post-Quanten-Kryptografie beginnt
  - Planung jetzt
  - Sukzessiver Umstieg im Laufe der nächsten Jahre (vgl. BSI-Empfehlungen)
- Verschlüsselung (KEM) solide, performant
- Digitale Signaturen: Situation je nach Einsatzzweck unklarer
- (Open-source) Libraries implementieren schon viel
  - Nicht immer reif für produktiven Einsatz
  - Auch Studis mit geringer Erfahrung konnten erfolgreich Demo-Szenarien aufbauen (insbesondere mit liboqs)

Fragen? Diskussion?

Danke für die Aufmerksamkeit!