IT-Sicherheit
Grundlagen

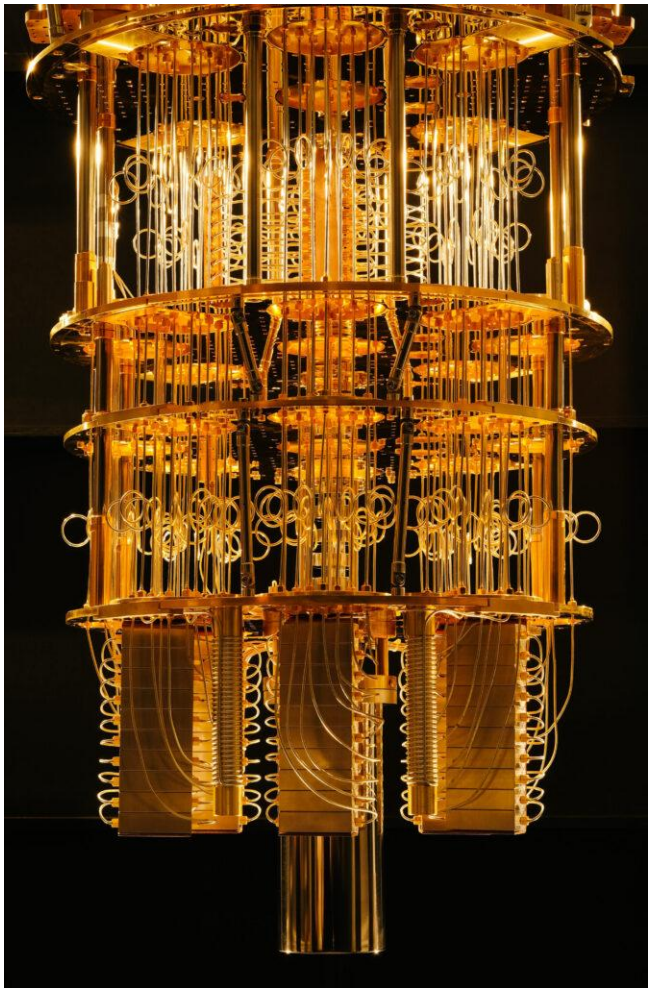
Computer Science Department
Systems Architecture Group
https://sar.informatik.hu-berlin.de

HTTPS, OpenVPN, TLS?

Quantensicheres Key Agreement

Wolf.Mueller@hu-berlin.de

Probleme: Quantencomputer



Quantencomputer rechnen grundlegend anders als Klassische.

Durch Benutzung von „gemischten Zuständen“ können Quantencomputer eine Vielzahl von Möglichkeiten in einem Zyklus in die Rechnung einbeziehen und das gemäß dem formulierten Quantenalgorithmus (resultierend in Randbedingungen an den Zustand) die richtige oder wahrscheinlichste Lösung „herausprojezieren“.

Aber: Bislang noch keine Quantencomputer die genügend Qubits kohärent über den nötigen Zeitraum der Rechnung bereitstellen können.

Aber: Wir müssen uns **jetzt vorbereiten!**
→ **P**ost **Q**uantum **C**ryptogaphy **PQC**

Quelle: https://www.higgs.ch/ein-quantencomputer-fuer-jedermann/25015/
bis zu 53-Qubit << 256 Bit (2019)

PQC: Grover / Bassard, …

Lov Kumar Grover. A fast quantum mechanical algorithm for database search.

In *Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing*, STOC '96, pages 212 219, New York, NY, USA, 1996. ACM.
https://doi.org/10.1145/237814.237866

Quantenalgorithmus zur Suche in unsortierter Datenbank mit N Einträgen in $\mathcal{O}\left(\sqrt{N}\right)$ Schritten und mit $\mathcal{O}(\log N)$ Speicherbedarf.

Quelle: https://www.dotquantum.io/en/quantum-biographies/

Gilles Brassard, Peter Høyer, and Alain Tapp. Quantum cryptanalysis of hash and claw-free functions. https://dl.acm.org/doi/10.5555/646387.690187

Quantenalgorithmus zur Suche von Hashkollisionen
Geburtstagsparadoxon $\mathcal{O}\left(2^{N/2}\right)$
$\Rightarrow \mathcal{O}\left(2^{N/3}\right)$

⇒ Erhöhung der Schlüssel- / Hashlängen in symmetrischen Verfahren!

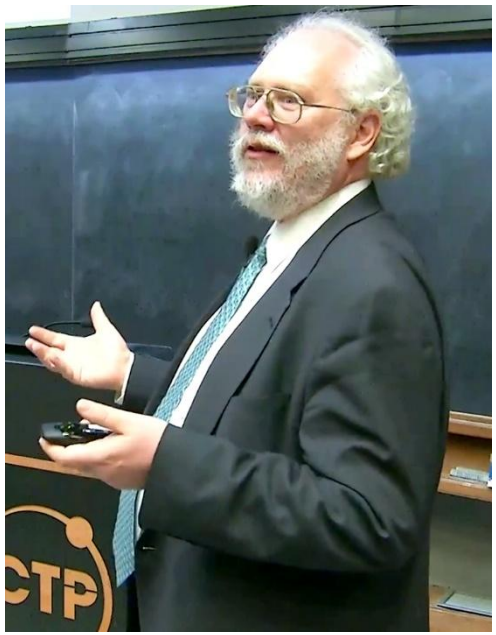
PQC: Shor

Peter Wiliston Shor. Algorithms for quantum computation: discrete logarithms and factoring.
In *Proceedings 35th Annual Symposium on Foundations of Computer Science*, SFCS '94, pages 124 134, Washington, DC, USA, Nov. 1994. IEEE https://doi.org/10.1109/SFCS.1994.365700

1996 Shor entwickelt Quantencomputer-Algorithmus, der Faktorisierungsproblem und (EC)DLP effizient

$$\mathcal{O}\left((\log N)^3\right)$$

löst.



Quelle: Von International Centre for Theoretical Physics – https://www.youtube.com/watch?v=J7HeDX_7Heg&t=7075, CC BY 3.0, https://commons.wikimedia.org/w/index.php?curid=75565986

⇒ **Angriffe mit polynomiellen Aufwand** mit Quantencomputer möglich gegen:
RSA, ECDSA, DH, ECDH, ElGamal …

BSI: Weitere Empfehlungen

https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/Broschueren/Kryptografie-quantensicher-gestalten.pdf

Wie viel Zeit bleibt für die Migration?

Um abzuschätzen, wann die Migration zu quantensicherer Kryptografie notwendig ist, ist die folgende Überlegung des theoretischen Physikers M. Mosca aus [*Mos15*] sehr anschaulich.

Sei

- x die Anzahl der Jahre, die die zu schützenden Daten abgesichert bleiben müssen,
- y die Anzahl der Jahre, die man benötigt, um das entsprechende System auf quantensichere Kryptografie umzustellen, sowie
- z die Anzahl der Jahre, die es noch dauert, bis Quantencomputer existieren, die die aktuell verwendete Kryptografie gefährden.

Dann gilt: Wenn $x + y > z$, dann haben Sie ein Problem!

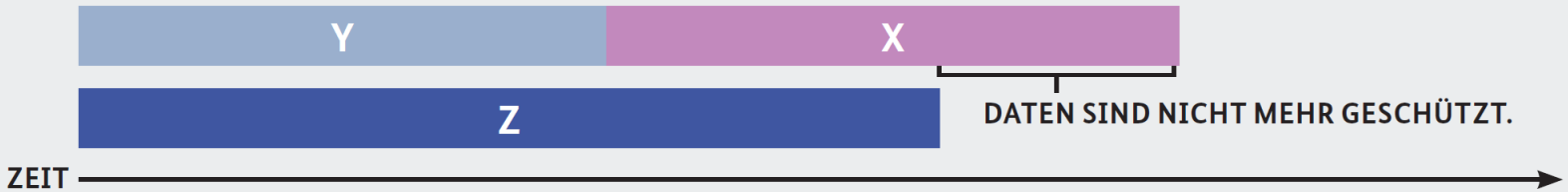


Abbildung: Veranschaulichung von „Moscas Theorem“

Frage: Wo sollte umgestellt werden?

Wo wäre es schlecht, wenn heute aufgezeichnete Daten in 10 (oder weniger) Jahren entschlüsselt werden können?

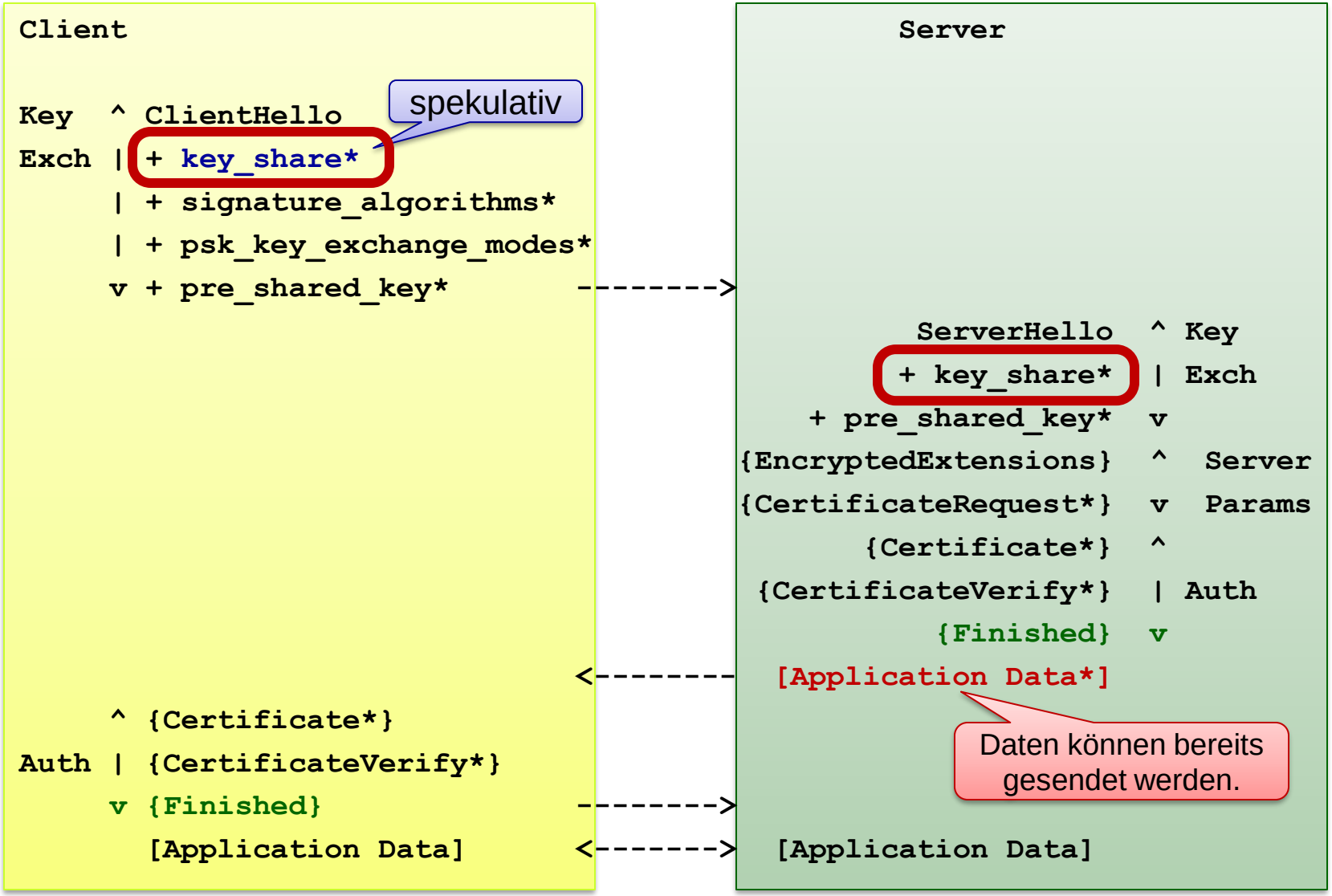
https://en.wikipedia.org/wiki/Harvest_now,_decrypt_later

- AGNES
- VPN
- LDAPS
- … ?

Bedrohung (Abwehrmöglichkeiten)**:**

- **Key-Exchange** (**dringend,** hybrider (PCQ+klassisch) Key Exchange)
- **Signatur-Verfahren** (später, quantensichere Signaturen)
- **Symmetrische Crypto** (256 Bit)

TLS 1.2 →TLS 1.3: Weniger Roudtrips



RFC 8446 https://tools.ietf.org/html/rfc8446#page-10

(Aktuell) sichere Ciphersuiten

Technische Richtlinie **TR-02102-2**

https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/TechnischeRichtlinien/TR02102/BSI-TR-02102-2.pdf

BSI TR-02102-2 "Kryptographische Verfahren: Verwendung von Transport Layer Security (TLS)"

Version: 2026-01

27.01.2026

Lesenswert!

TR-02102-2: Änderungshistorie

Version	*Datum*	*Beschreibung*
2019-01	22.02.2019	Anpassung der Verwendungszeiträume, Empfehlung von TLS 1.3, Empfehlung von PSK-Cipher-Suiten aus RFC 8442, Empfehlung des CCM-Modus
2020-01	28.02.2020	Anpassung der Verwendungszeiträume, Abkündigung von HMAC SHA-1
2021-01	12.03.2021	Anpassung der Verwendungszeiträume
2022-01	24.01.2022	Anpassung der Verwendungszeiträume, Empfehlung der elliptischen Kurve secp521r1
2023-01	17.01.2023	Anhebung des Sicherheitsniveaus auf 120 Bit, Anpassung der Verwendungszeiträume
2024-01	29.02.2024	Anpassung der Verwendungszeiträume, Ablauf der Empfehlung von DSA und von DHE-Cipher-Suiten ab Ende 2029
2025-01	21.01.2025	Anpassung der Verwendungszeiträume
2026-01	**27.01.2026**	Anpassung der Verwendungszeiträume, **Abkündigung von RSA-Signaturen mit PKCS #1 v1.5 Padding**, **Migrationsfristen zu quantensicheren Schlüsseleinigungsverfahren**, **Ablauf der Empfehlung von TLS 1.2 ab Ende 2031**

TR-02102-2: Empfohlene TLS-Versionen

TLS-Version	Referenz	Verwendung bis
TLS 1.3	**[RFC 8446]**	**2032+**
TLS 1.2	[RFC 5246]	2031

- **TLS 1.3 sollte bevorzugt verwendet werden.**
- TLS 1.2 wird nur noch bis Ende 2031 empfohlen, da nach derzeitigem Kenntnisstand **keine quantensicheren Schlüsseleinigungsverfahren für TLS 1.2 standardisiert werden**.
- TLS 1.0 und TLS 1.1 werden nicht empfohlen (siehe auch [RFC 8996]).
- SSL v2 ([SSLv2]) und SSL v3 ([SSLv3]) werden nicht empfohlen (siehe auch [RFC 6176] und [RFC 7568]).

TR-02102-2: Empfohlene DH-Gruppen TLS 1.3

Diffie-Hellman-Gruppe	IANA-Nr.	Referenz	Verwendung bis
secp256r1	23	[RFC 8422]	2031
secp384r1	24	[RFC 8422]	2031
secp521r1	25	[RFC 8422]	2031
brainpoolP256r1tls13	31	[RFC 8734]	2031
brainpoolP384r1tls13	32	[RFC 8734]	2031
brainpoolP512r1tls13	33	[RFC 8734]	2031
ffdhe3072	257	[RFC 7919]	2031

Bei (EC)DHE handelt es sich um **klassische Schlüsseleinigungsverfahren**, die **nicht quantensicher** sind. Daher wird die alleinige Nutzung der Diffie-Hellman-Gruppen in Tabelle 10 nur noch bis Ende 2031 empfohlen. Das BSI beabsichtigt, die **quantensicheren hybriden Schlüsseleinigungsverfahren SecP256r1MLKEM768** und **SecP384r1MLKEM1024** aus dem Internet-Draft unter https://datatracker.ietf.org/doc/draft-ietf-tls-ecdhe-mlkem/ zu empfehlen, sobald der zugehörige RFC verabschiedet wurde.

https://datatracker.ietf.org/doc/draft-ietf-tls-ecdhe-mlkem/

Voraussetzungen: TLS-Bibliothek

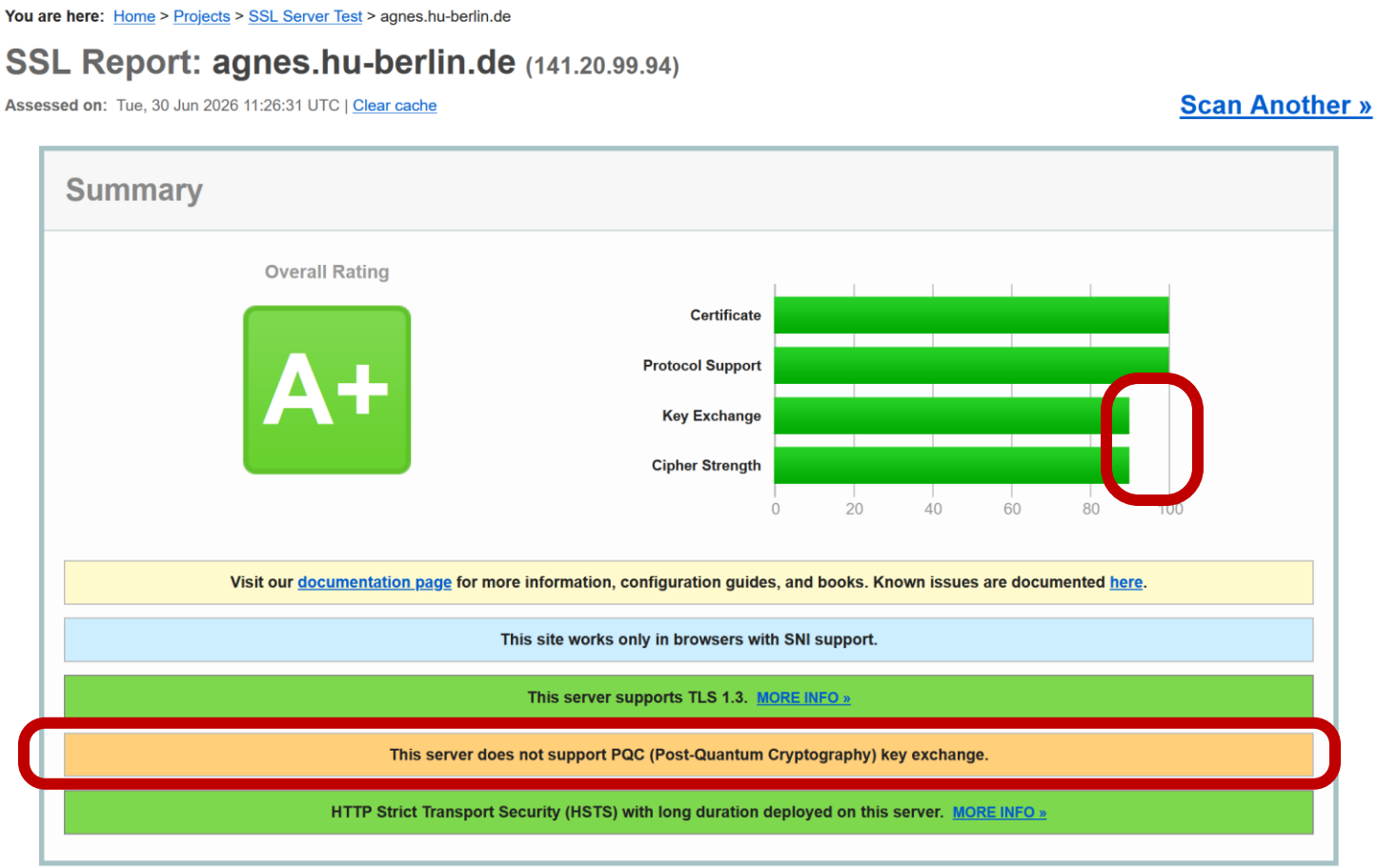
Quelle: https://ki-tools.hu-berlin.de/chat/ „ Mache mir eine Übersicht, welche verbreiteten TLS-Bibliotheken ab welcher Version SecP384r1MLKEM1024 für den Key Exchange unterstützen.“

TLS-Bibliotheken: SecP384r1MLKEM1024-Unterstützung

Bibliothek	Version	Anmerkungen
OpenSSL	3.5.0 ✓	Erste stabile Version mit hybriden PQC-Gruppen; SecP384r1MLKEM1024 als Codepoint 0x0205
BoringSSL	Commit `5f6c5a2` (März 2025)	Google's Fork, Basis für Chrome/Chromium; Gruppe aktiv in internen Tests
AWS-LC	1.49.0 ✓	AWS-Fork von BoringSSL, PQC-Gruppen übernommen
GnuTLS	3.8.9 ✓	Über `GNUTLS_GROUP_SECP384R1_MLKEM1024`; experimentell in 3.8.8
wolfSSL	5.7.6 ✓	`--enable-kyber` erforderlich; hybride Gruppen seit 5.7.0 sukzessive erweitert
mbed TLS	3.6.3 (LTS) / 3.7.0	PQC-Unterstützung über PSA-Crypto-API; ML-KEM erst ab 3.6.2, hybride Gruppen in 3.7
LibreSSL	*nicht vorhanden*	Explizit keine PQC-Implementierung geplant (Sicherheitsbedenken bzgl. NIST-Prozess)
s2n-tls	1.5.12	AWS-TLS-Wrapper, nutzt AWS-LC bzw. OpenSSL 3.5+ als Backend

Beispiel: AGNES (soweit bekannt …)

https://www.ssllabs.com/ssltest/analyze.html?d=agnes.hu-berlin.de



Nebenbefund: **AGNES**

Additional Certificates (if supplied)

Certificates provided	4 (7564 bytes)
Chain issues	**Incorrect order, Extra certs**

#2

Subject	agnes.hu-berlin.de Fingerprint SHA256: 5a7400ce2c0657941fdad8c2024d9d45501307f50e0146b8bf357755278cf167 Pin SHA256: 3l3XB8y1TdsjlEgS17AaBtWQryQ5VFbNY32l6URV5vg=
Valid until	Thu, 01 Oct 2026 08:48:56 UTC (expires in 3 months)
Key	EC 256 bits
Issuer	GEANT TLS ECC 1
Signature algorithm	SHA384withECDSA

Vermutung: Server Certificate wird nochmals als Bestandteil der Cerificate Chain ausgeliefert?

Ansatz: Vergleich
https://www.ssllabs.com/ssltest/analyze.html?d=agnes.hu-berlin.de
https://www.ssllabs.com/ssltest/analyze.html?d=sarwiki.informatik.hu-berlin.de

Test: sarwiki.informatik.hu-berlin.de

SSL Report: sarwiki.informatik.hu-berlin.de (141.20.23.62)

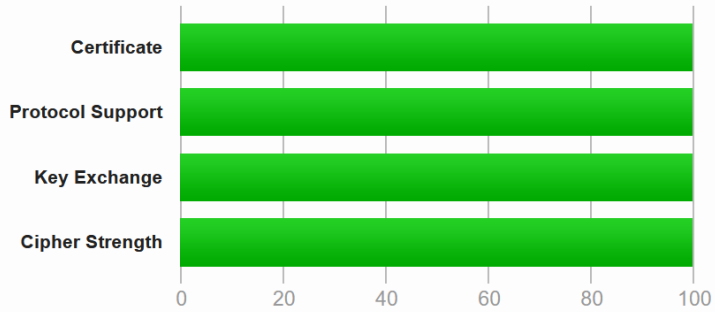
Assessed on: Tue, 30 Jun 2026 12:24:47 UTC | **HIDDEN** | Clear cache

Scan Another »

Summary

Overall Rating

A+



Visit our documentation page for more information, configuration guides, and books. Known issues are documented here.

This server supports TLS 1.3. MORE INFO »

This server supports PQC (Post-Quantum Cryptography) key exchange.

HTTP Strict Transport Security (HSTS) with long duration deployed on this server. MORE INFO »

Input: BSI, Google, AI?

https://configurator.tlsref.org/

https://configurator.tlsref.org/#server=apache&version=2.4.67&config=modern&openssl=3.5.6%207&ocsp&guideline=6.0

Apache 2.4.67, OpenSSL 3.5.6 7, modern config

```
# this configuration requires mod_ssl and mod_socache_shmcb
<VirtualHost *:443>
    SSLEngine on
    SSLCertificateFile      /path/to/signed_cert_and_intermediate_certs
    SSLCertificateKeyFile   /path/to/private_key

    # enable HTTP/2, if available
    Protocols h2 http/1.1
</VirtualHost>

# modern configuration
SSLProtocol             -all +TLSv1.3
SSLOpenSSLConfCmd       Curves X25519MLKEM768:X25519:prime256v1:secp384r1
SSLCipherSuite          TLSv1.3
TLS_AES_128_GCM_SHA256:TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA256
SSLHonorCipherOrder     off
SSLSessionTickets       off

SSLUseStapling On
SSLStaplingCache "shmcb:logs/ssl_stapling(32768)"
```

AI: Was ist in Apache möglich?

```
root@sarwiki:~# openssl list  -kem-algorithms
  { 1.2.840.113549.1.1.1, 2.5.8.1.1, RSA, rsaEncryption } @ default
  { 1.2.840.10045.2.1, EC, id-ecPublicKey } @ default
  { 1.3.101.110, X25519 } @ default
  { 1.3.101.111, X448 } @ default
  { 2.16.840.1.101.3.4.4.1, id-alg-ml-kem-512, ML-KEM-512, MLKEM512 } @ default
  { 2.16.840.1.101.3.4.4.2, id-alg-ml-kem-768, ML-KEM-768, MLKEM768 } @ default
  { 2.16.840.1.101.3.4.4.3, id-alg-ml-kem-1024, ML-KEM-1024, MLKEM1024 } @ default
  X25519MLKEM768 @ default
  X448MLKEM1024 @ default
  SecP256r1MLKEM768 @ default
  SecP384r1MLKEM1024 @ default

SSLProtocol         -all +TLSv1.3
SSLOpenSSLConfCmd   Curves X25519MLKEM768:X448MLKEM1024:SecP256r1MLKEM768:SecP384r1MLKEM1024
```

root@sarwiki:~# systemctl restart apache2.service

startet nicht ☹

Fehlersuche

Was genau verwendend der Webserver?
Apache2 → mod_ssl.so → libssl

```
ldd /usr/lib/apache2/modules/mod_ssl.so | grep libssl
strings /usr/lib/x86_64-linux-gnu/libssl.so.3 | grep  -i 'MLKEM‘
```

```
MLKEM512
SecP256r1MLKEM768
X25519MLKEM768
SecP384r1MLKEM1024
?*X25519MLKEM768 / ?*X25519:?secp256r1 /
?X448:?secp384r1:?secp521r1 / ?ffdhe2048:?ffdhe3072
```

modern configuration++

```
# TLS 1.3 only
SSLProtocol         -all +TLSv1.3
# PQC-Hybrid only
SSLOpenSSLConfCmd Groups SecP384r1MLKEM1024:X25519MLKEM768:SecP256r1MLKEM768
```

Läuft, **ich** kann mich verbinden:

```
openssl s_client -connect sarwiki.informatik.hu-berlin.de:443
…
Peer signing digest: SHA384
Peer signature type: ecdsa_secp384r1_sha384
Negotiated TLS1.3 group: X25519MLKEM768
```

Aktuelle Browser gehen auch!

Probleme:

- **ICINGA-Warnung** „host down“ (OpenSSL auf ICINGA-Rechner kann kein PQC)
- **SSLLABS** „Webserver nicht gefunden“ (TLS 1.3 only nicht unterstützt ☹)

intermediate configuration++

```
# Intermediate
# TLS 1.2 oder 1.3
SSLProtocol             -all +TLSv1.2 +TLSv1.3

# nur 256bit cipher
SSLCipherSuite      ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-
CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-POLY1305
SSLCipherSuite      TLSv1.3 TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA256

# nur hybrid oder klassische EC-Gruppen (>= 384bit)
SSLOpenSSLConfCmd Groups
SecP384r1MLKEM1024:X25519MLKEM768:SecP256r1MLKEM768:secp521r1:secp384r1
```

Aktuelle Browser gehen auch!

- ICINGA OK
- SSLLABS OK

Interoperabilität: OK, aber individuell

https://www.ssllabs.com/ssltest/analyze.html?d=sarwiki.informatik.hu-berlin.de

Handshake Simulation	
Android 5.0.0	Server sent fatal alert: handshake_failure
Android 6.0	Server sent fatal alert: handshake_failure
Chrome 49 / XP SP3	Server sent fatal alert: handshake_failure
Firefox 31.3.0 ESR / Win 7	Server sent fatal alert: handshake_failure
Safari 6 / iOS 6.0.1	Server sent fatal alert: handshake_failure
Safari 7 / iOS 7.1 R	Server sent fatal alert: handshake_failure
Safari 7 / OS X 10.9 R	Server sent fatal alert: handshake_failure
Safari 8 / iOS 8.4 R	Server sent fatal alert: handshake_failure
Safari 8 / OS X 10.10 R	Server sent fatal alert: handshake_failure

Benchmark „Was kostet der Spaß?“

```
root@sarwiki:~# openssl s_time -help
Usage: s_time [options]

Connection options:
 -connect val        Where to connect as post:port (default is
localhost:4433)
 -new                Just time new connections
 -reuse              Just time connection reuse
 -bugs               Turn on SSL bug compatibility
 -cipher val         TLSv1.2 and below cipher list to be used
 -ciphersuites val   Specify TLSv1.3 ciphersuites to be used
 -tls1               Just use TLSv1.0
 -tls1_1             Just use TLSv1.1
 -tls1_2             Just use TLSv1.2
 -tls1_3             Just use TLSv1.3
 -verify +int        Turn on peer certificate verification, set depth
 -time +int          Seconds to collect data, default 30
 -www val            Fetch specified page from the site
```

Keine Option um -curves oder -groups zu setzen ☹

Benchmark: Kann die AI das mal nachrüsten?

```
/* * sslbench.c - TLS handshake benchmark with configurable group/curve * Similar to openssl s_time but with
explicit group selection for key agreement */

static void usage(const char *prog)
{
    fprintf(stderr,
        "Usage: %s -connect host:port [options]\n"
        "Options:\n"
        "  -connect host:port    Target server (required)\n"
        "  -groups list          Colon-separated group names (e.g., 'X25519:P-256:P-384')\n"
        "  -curves list          Alias for -groups (TLS 1.2 legacy naming)\n"
        "  -cipher list          Cipher list string\n"
        "  -ciphersuites list    TLS 1.3 ciphersuites\n"
        "  -time seconds         Benchmark duration (default: 30)\n"
        "  -new                  Use new session for each handshake\n"
        "  -reuse                Reuse session (default)\n"
        "  -verify depth         Enable certificate verification\n"
        "  -CAfile file          CA certificate file\n"
        "  -cert file            Client certificate\n"
        "  -key file             Client private key\n"
        "  -tls1_3               Force TLS 1.3\n"
        "  -tls1_2               Force TLS 1.2\n"
        "  -nbio                 Use non-blocking IO\n"
        "  -www page             HTTP GET request path\n"
        "  -bytes n              Read n bytes after handshake\n"
        , prog);
    exit(1);
}
```

Benchmark: Skepsis, funktioniert das auch?

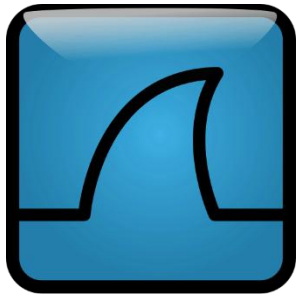
```
sslbench: sslbench.c
        gcc -O2 -o sslbench sslbench.c -lssl –lcrypto
```

… baut.

```
./sslbench -time 30 \
-connect sarwiki.informatik.hu-berlin.de:443 \
-new \
-tls1_3 \
-groups SecP384r1MLKEM1024
```

… läuft …

… (scheinbar) korrekt, Inspektion mit Wireshark



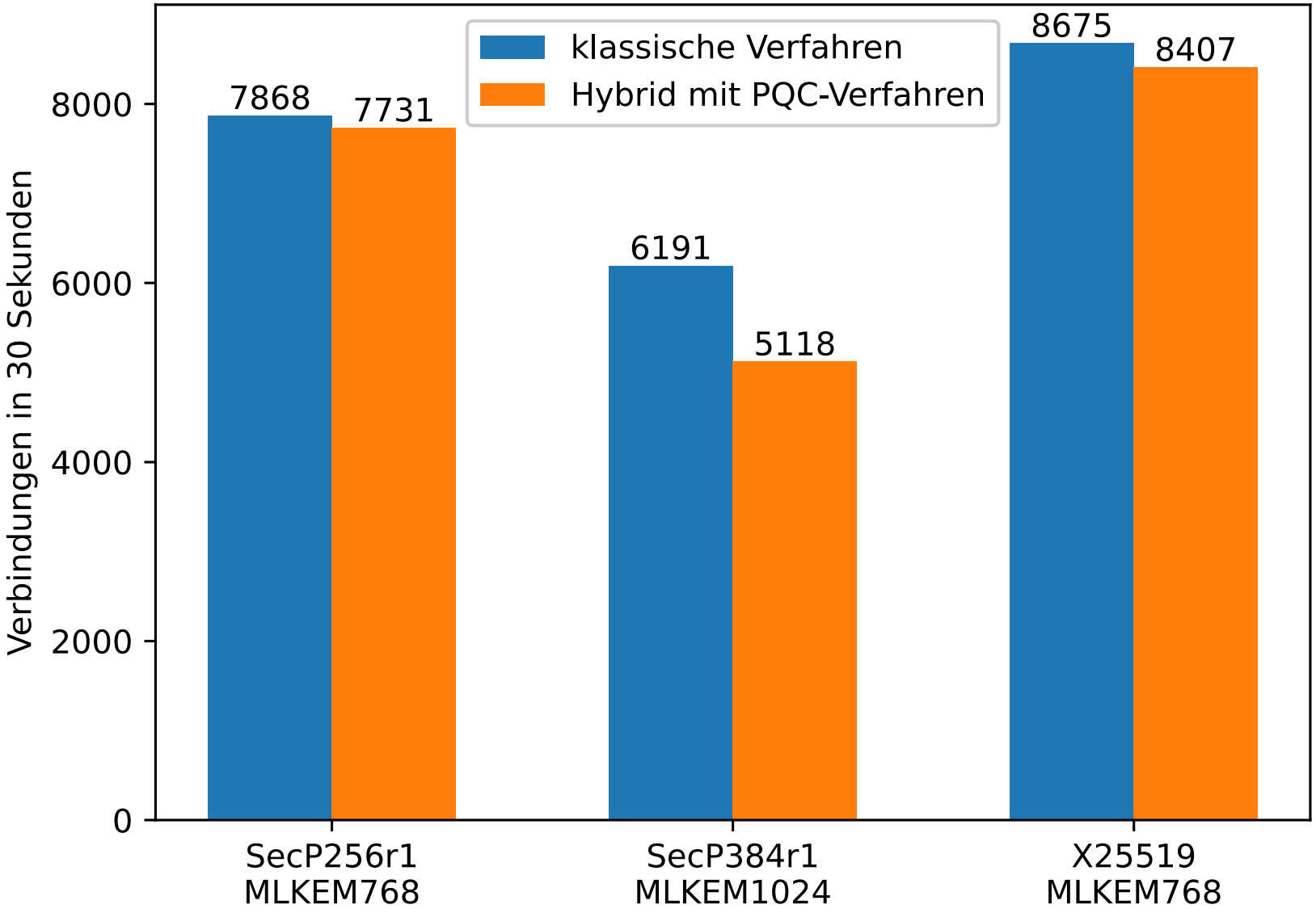
Benchmark: Messung

```
#!/bin/bash
#zu evaluierende Gruppen
Groups="SecP384r1MLKEM1024:X25519MLKEM768:SecP256r1MLKEM768:secp384r1:X25519:prime256v1"

(\
for G in ${Groups//:/ }; do
   echo ===== $G =====
   ./sslbench -time 30 -connect sarwiki.informatik.hu-berlin.de:443 -new -tls1_3 -groups $G
|\
   sed -e 's@^\([0-9]* connections\)@['$G'] \1@g'
done ) | tee bench.log

awk '/^\[/ {print $0}' bench.log | sort -f
```

Benchmark: Plot



Benchmark: Ergebnisse

@Intel(R) Xeon(R) CPU E5-2670 0 @ 2.60GHz, 1 core

```
root@sarwiki:~# awk '/^\[/ {print $0}' bench.log | sort -f
[prime256v1] 7868 connections in 30.00s; 262.24 connections/user sec, 3.212 ms avg
[SecP256r1MLKEM768] 7731 connections in 30.00s; 257.66 connections/user sec, 3.380 ms avg

[secp384r1] 6191 connections in 30.00s; 206.36 connections/user sec, 4.348 ms avg
[SecP384r1MLKEM1024] 5118 connections in 30.00s; 170.59 connections/user sec, 5.417 ms avg

[X25519] 8675 connections in 30.00s; 289.16 connections/user sec, 2.940 ms avg
[X25519MLKEM768] 8407 connections in 30.19s; 278.44 connections/user sec, 3.045 ms avg
```

- Da ist aus meiner Sicht der Mehraufwand hinsichtlich Latenz durchaus akzeptabel.
- Natürlich sind aber auch die Pakete größer, aber Speed ist jetzt kein Showstopper 😉

Feedback

- Fragen?
- Bemerkungen?
- Hinweise?

PQC@OpenVPN

vielleicht als Thema im **IT-Security Workshop 2026**

https://sarwiki.informatik.hu-berlin.de/Main_Page

```
--tls-groups SecP384r1MLKEM1024:X25519MLKEM768:SecP256r1MLKEM768
ldd /sbin/openvpn | grep ssl
   libssl.so.3 => /lib64/glibc-hwcaps/x86-64-v3/libssl.so.3.5.0 (0x00007fb84c5fe000)
Alternativ klassisch --tls-groups secp521r1
```