PQC/TLS-Handshake

...

von
Kay Schmelzer

TLS - Handshake

-Schlüsselaushandlung

-Verschlüsselung der Kommunikation mit Integritätsprüfung

-Authentifizierung über Zertifikate

OpenSSL - Server und Client

openssl req -new -x509 -newkey rsa:2048 -nodes -keyout key.pem -out cert.pem -days 365

openssl s_server -accept 4433 -cert cert.pem -key key.pem

openssl s_client -connect localhost:4433

Wireshark

```
TCP        64 63643 → 1234 [ACK] Seq=1 Ack=1 Win=65475 Len=0
TLSv1.3  1457 Client Hello
TCP        64 1234 → 63643 [ACK] Seq=1 Ack=1394 Win=64082 Len=0
TLSv1.3  2655 Server Hello, Change Cipher Spec, Application Data, Application Data, Application Data, Application Data
TCP        64 63643 → 1234 [ACK] Seq=1394 Ack=2592 Win=62884 Len=0
TLSv1.3   144 Change Cipher Spec, Application Data
TCP        64 1234 → 63643 [ACK] Seq=2592 Ack=1474 Win=64002 Len=0
TLSv1.3   319 Application Data
TCP        64 63643 → 1234 [ACK] Seq=1474 Ack=2847 Win=62629 Len=0
TLSv1.3   319 Application Data
TCP        64 63643 → 1234 [ACK] Seq=1474 Ack=3102 Win=62374 Len=0
TLSv1.3    88 Application Data
TCP        64 1234 → 63643 [ACK] Seq=3102 Ack=1498 Win=63978 Len=0
TCP        64 63643 → 1234 [FIN, ACK] Seq=1498 Ack=3102 Win=62374 Len=0
TCP        64 1234 → 63643 [ACK] Seq=3102 Ack=1499 Win=63978 Len=0
TLSv1.3    88 Application Data
TCP        64 63643 → 1234 [ACK] Seq=1499 Ack=3126 Win=62350 Len=0
TCP        64 1234 → 63643 [FIN, ACK] Seq=3126 Ack=1499 Win=63978 Len=0
TCP        64 63643 → 1234 [ACK] Seq=1499 Ack=3127 Win=62350 Len=0
TCP        72 63645 → 1234 [SYN] Seq=0 Win=65475 Len=0 MSS=65475 SACK_PERM
TCP        72 1234 → 63645 [SYN, ACK] Seq=0 Ack=1 Win=65475 Len=0 MSS=65475 SACK_PERM
```

Wireshark - keylogfile

openssl s_client -connect localhost:1235 -tls1_3 -keylogfile keylogfile.log

```
TCP         64 60034 → 1234 [ACK] Seq=1 Ack=1 Win=65475 Len=0
TLSv1.3   1457 Client Hello
TCP         64 1234 → 60034 [ACK] Seq=1 Ack=1394 Win=64082 Len=0
TLSv1.3   2655 Server Hello, Change Cipher Spec, Encrypted Extensions, Certificate, Certificate Verify, Finished
TCP         64 60034 → 1234 [ACK] Seq=1394 Ack=2592 Win=62884 Len=0
TLSv1.3    144 Change Cipher Spec, Finished
TCP         64 1234 → 60034 [ACK] Seq=2592 Ack=1474 Win=64002 Len=0
TLSv1.3    319 New Session Ticket
TCP         64 60034 → 1234 [ACK] Seq=1474 Ack=2847 Win=62629 Len=0
TLSv1.3    319 New Session Ticket
TCP         64 60034 → 1234 [ACK] Seq=1474 Ack=3102 Win=62374 Len=0
TLSv1.3    108 Application Data
TCP         64 1234 → 60034 [ACK] Seq=3102 Ack=1518 Win=63958 Len=0
```

Das "store now and decrypt later"-Szenario

-klassische Schlüsselaushandlung aktuell noch sicher

-Bedrohung durch künftige Verbreitung von Quantencomputern

TLS - Schlüsselaushandlung

-Verschiedene Algorithmen für die Berechnung der Schlüssel

-In der Kommunikation gibt es private und öffentliche Schlüssel

-Berechnung des “Shared Secrets” privatem und öffentlichem Schlüssel

-Shared Secret -> Ableitung der Sitzungsschlüssel über z.B. HKDF

ECDH

-**Elliptic-curve Diffie–Hellman**

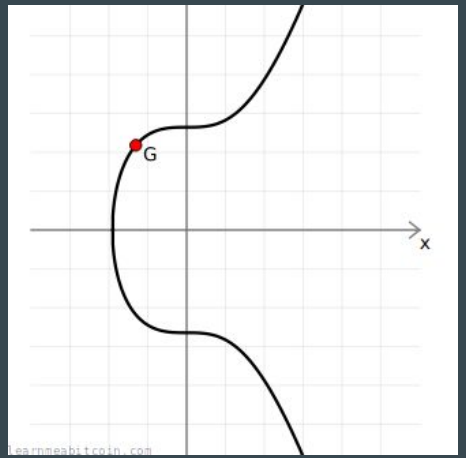
-**z.B.** X25519 und SecP256r1 sind solche Kurven

-Client und Server bestimmten jeweils einen zufälligen privaten Schlüssel

-G = Zufälliger Punkt auf der Kurve, gleich für Server und Client

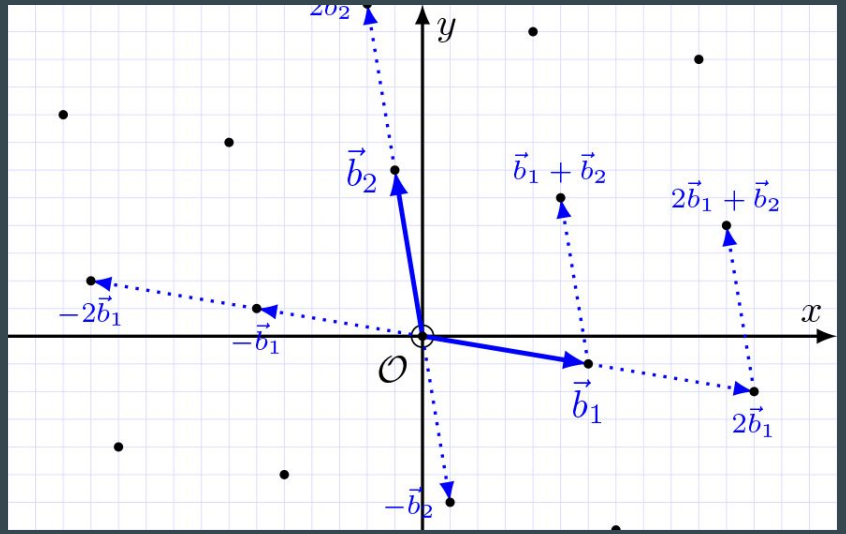
-öffentlicher Schlüssel = privater Schlüssel * G

-Berechnung des Shared Secrets:S=aB=a(bG)=b(aG)=bA



ML-KEM

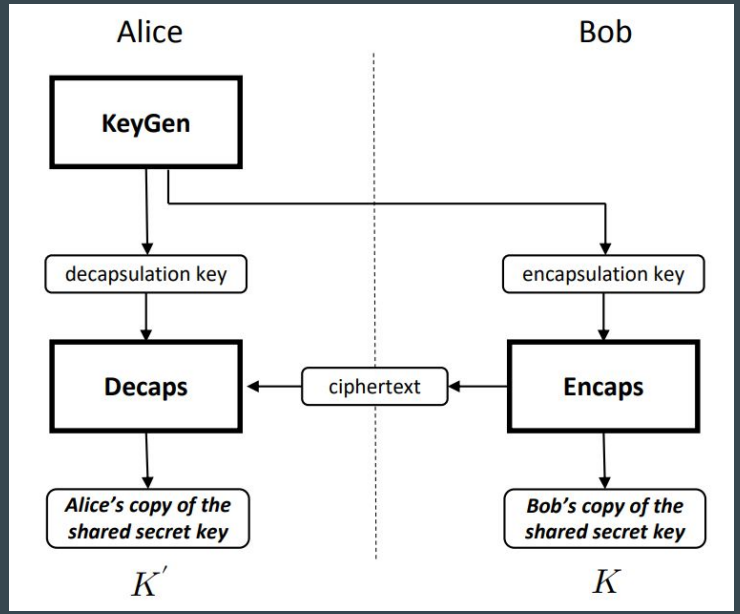
-Ist ein Key Encapsulation Mechanism
-Basiert auf Gitter im mehrdimensionalen Raum (z.B. n =256)
-Schlüsselpaare deutlich größer als für klassische Verfahren (teilweise mehrere KB)
-Schlüsselberechnungen nutzen Matrizen, Vektoren mit Dimension n und Polynome (n-1)-ten Grades



ML-KEM

-Server generiert die Schlüssel
-Öffentlicher Schlüssel wird von Client um einen Schlüsselwert erweitert und mit einem Störfaktor an den Server gesandt
-Server kann mit seinem privaten Schlüssel den Schlüsselwert ermitteln
-Beide Partien haben nun Zugang zum Shared Secret

-Die Ermittlung des Schlüsselwertes ist ein simples lineares Gleichungssystem jedoch durch den eingebauten Störfaktor selbst für Quantencomputer nicht realistisch lösbar



Hybride Schlüsselaushandlungverfahren

-Parallele Nutzung eines klassischen und eines quantensicheren Verfahrens

-Schlüsselpaare werden als Konkationation übertragen

-Das Shared Secret ergibt sich aus den Shared Secrets beider Algorithmen

-Doppelt sicher selbst bei Implementationsfehlern oder -schwächen

Performance

-SecP256r1

-MLKEM768

-X25519MLKEM768

-SecP256r1MLKEM768

-SecP384r1MLKEM1024

Performance - CPU

Erwartung:

SecP256r1 < MLKEM768 < X25519MLKEM768 ≈ SecP256r1MLKEM768 < SecP384r1MLKEM1024

Performance - CPU

Erwartung:

SecP256r1 < MLKEM768 < X25519MLKEM768 ≈ SecP256r1MLKEM768 < SecP384r1MLKEM1024

Tatsächlich:

MLKEM768 < X25519MLKEM768 ≈ SecP256r1MLKEM768 < SecP384r1MLKEM1024 < SecP256r1

19.1% 20,0% 20,4% 21,3% 22,5%

Performance - Zeit in ms

SecP256r1	MLKEM768	X25519MLKEM768	SecP256r1MLKEM768	SecP384r1MLKEM1024
2,819	2,601	3,165	2,887	4,68
2,006	2,194	2,171	2,391	4,648
2,002	2,364	2,22	2,136	4,021
1,99	2,118	2,276	2,245	4,346
2,102	2,164	2,161	2,195	4,433
1,997	2,149	2,232	2,167	4,223
2,163	2,189	2,016	2,28	3,878
2,043	2,104	2,101	2,602	4,392
2,013	2,015	2,22	2,139	4,391
2,186	1,879	2,139	2,373	4,407

Performance - Zeit in ms

	SecP256r1	< MLKEM768	< X25519MLKEM768	≈ SecP256r1MLKEM768	< SecP384r1MLKEM1024
∅	2,1321	2,1777	2,2701	2,3415	4,3419

Interoperabilität

-OpenSSL

-GnuTLS

-wolfSSL

```
D:\Uni\IT-Security>openssl s_server -accept 4433 -cert cert.pem -key key.pem -www
```

-Firefox

->

Interoperabilität - OpenSSL

-Ab 3.5.x

-openssl list -tls1_3 -tls-groups

-openssl s_server -accept 4433 -cert cert.pem -key key.pem -tls1_3 -groups "X25519MLKEM768"

-openssl s_client -connect localhost:4433 -tls1_3 -groups "X25519MLKEM768"

Interoperabilität - GnuTLS

-Laut Dokumentation:

- -Ab 3.8.8: “experimental support” für X25519MLKEM768, SecP256r1MLKEM768
- -Ab 3.8.9: auch SecP384r1MLKEM1024

Interoperabilität - wolfSSL

Protocol Support

(D)TLS 1.3, MQTTv5, and MQTT-SN

Supported Groups Extension Codepoints

- ML_KEM_512
- ML_KEM_768
- ML_KEM_1024 (CNSA 2.0 compliant)
- P256_ML_KEM_512 (hybrid with FIPS 140-3)
- P384_ML_KEM_768 (hybrid with FIPS 140-3)
- P521_ML_KEM_1024 (hybrid with FIPS 140-3; CNSA 2.0 compliant)
- P256_ML_KEM_768 (hybrid with FIPS 140-3)
- P384_ML_KEM_1024 (hybrid with FIPS 140-3; CNSA 2.0 compliant)
- X25519_ML_KEM_768 (hybrid with FIPS 140-3)

Interoperabilität - Firefox mit X25519MLKEM768

-https://localhost:port

```
TCP        44 56164 → 1234 [ACK] Seq=1 Ack=1 Win=65495 Len=0
TLSv1.3  1944 Client Hello (SNI=localhost)
TCP        44 1234 → 56164 [ACK] Seq=1 Ack=1901 Win=63595 Len=0
TLSv1.3  2619 Server Hello, Change Cipher Spec, Encrypted Extensions, Certificate, Certificate Verify, Finished
TCP        44 56164 → 1234 [ACK] Seq=1901 Ack=2576 Win=62920 Len=0
TLSv1.3   108 Change Cipher Spec, Finished
TCP        44 1234 → 56164 [ACK] Seq=2576 Ack=1965 Win=63531 Len=0
TLSv1.3   283 New Session Ticket
HTTP      523 GET / HTTP/1.1
```

```
▾ Extension: key_share (len=1327) X25519MLKEM768, x25519, secp256r1
    Type: key_share (51)
    Length: 1327
  ▾ Key Share extension
      Client Key Share Length: 1325
    ▾ Key Share Entry: Group: X25519MLKEM768, Key Exchange length: 1216
        Group: X25519MLKEM768 (4588)
        Key Exchange Length: 1216
        Key Exchange […]: 28d057deec7c8f9047a5c678c356c786e9a22a8a54ac3302bb6111a9981c2b
    ▾ Key Share Entry: Group: x25519, Key Exchange length: 32
        Group: x25519 (29)
        Key Exchange Length: 32
        Key Exchange: 578a53f997a39e15afeeb7a70102a2ac6b7cabfda64c472cc9c107dbc99a915b
    ▾ Key Share Entry: Group: secp256r1, Key Exchange length: 65
        Group: secp256r1 (23)
        Key Exchange Length: 65
        Key Exchange: 04f661de7899917ee51ddf3eedd196b2e6358aa49f3c51e8a093bb7bb796b2af1
```

Interoperabilität - Firefox mit SecP384r1MLKEM1024

Server-SecP384r1MLKEM1024

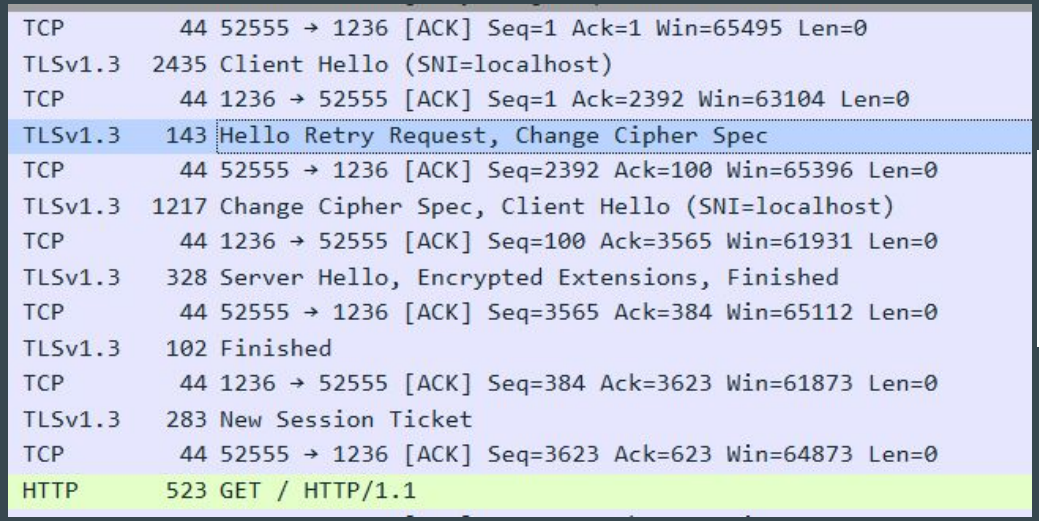
```
Using default temp DH parameters
ACCEPT
E8690000:error:0A000065:SSL routines:final_key_share:no suitable key share:..\ssl\statem\extensions.c:1467:
E8690000:error:0A000065:SSL routines:final_key_share:no suitable key share:..\ssl\statem\extensions.c:1467:
```

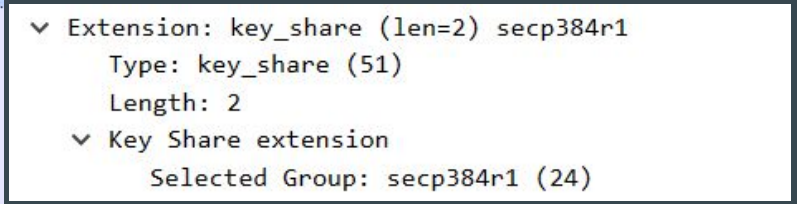
TCP	44	56876 → 1236 [ACK] Seq=1 Ack=1 Win=65495 Len=0
TLSv1.2	1944	Client Hello (SNI=localhost)
TCP	44	1236 → 56876 [ACK] Seq=1 Ack=1901 Win=63595 Len=0
TLSv1.2	51	Alert (Level: Fatal, Description: Handshake Failure)
TCP	44	56876 → 1236 [ACK] Seq=1901 Ack=8 Win=65488 Len=0
TCP	44	56876 → 1236 [FIN, ACK] Seq=1901 Ack=8 Win=65488 Len=0
TCP	44	1236 → 56876 [ACK] Seq=8 Ack=1902 Win=63595 Len=0
TCP	44	1236 → 56876 [FIN, ACK] Seq=8 Ack=1902 Win=63595 Len=0

-Ebenso für SecP256r1MLKEM768

Interoperabilität - Firefox mit SecP384r1

-Unterstützter nicht anfangs gelisteter Schlüsselalgorithmus





Interoperabilität - Firefox mit SecP384r1

-Neues Client Hello

```
TCP        44 52555 → 1236 [ACK] Seq=1 Ack=1 Win=65495 Len=0
TLSv1.3  2435 Client Hello (SNI=localhost)
TCP        44 1236 → 52555 [ACK] Seq=1 Ack=2392 Win=63104 Len=0
TLSv1.3   143 Hello Retry Request, Change Cipher Spec
TCP        44 52555 → 1236 [ACK] Seq=2392 Ack=100 Win=65396 Len=0
TLSv1.3  1217 Change Cipher Spec, Client Hello (SNI=localhost)
TCP        44 1236 → 52555 [ACK] Seq=100 Ack=3565 Win=61931 Len=0
TLSv1.3   328 Server Hello, Encrypted Extensions, Finished
TCP        44 52555 → 1236 [ACK] Seq=3565 Ack=384 Win=65112 Len=0
TLSv1.3   102 Finished
TCP        44 1236 → 52555 [ACK] Seq=384 Ack=3623 Win=61873 Len=0
TLSv1.3   283 New Session Ticket
TCP        44 52555 → 1236 [ACK] Seq=3623 Ack=623 Win=64873 Len=0
HTTP      523 GET / HTTP/1.1
```

```
▾ Extension: key_share (len=103) secp384r1
    Type: key_share (51)
    Length: 103
  ▾ Key Share extension
      Client Key Share Length: 101
    ▾ Key Share Entry: Group: secp384r1, Key Exchange length: 97
        Group: secp384r1 (24)
        Key Exchange Length: 97
        Key Exchange: 0437ea5f5090250649c6165ed6c1b494a1eda3907612a66fffbc813
```

Quantensichere Zertifikate

-Scheinbar noch keine öffentlichen CA's (Certificate Authority) für PCQ-Zertifikate

-Es gibt bereits Lösungen für quantensichere Zertifikate durch private CA's

-Einige Firmen testen und arbeiten bereits mit privaten Zertifikaten

-Ein bereits anerkannter Standard für PQC-Zertifikate ist der Module Lattice–Based Digital Signature Algorithm (ML-DSA)

Hybride Zertifikate

-Es gibt noch keine einsatzfähigen hybriden Zertifikate (stand 25.08.2025)

-Wird bereits stark diskutiert und als potentielle Lösung angesehen

-Idee: Jedes Zertifikat hat 2 Signaturen und 2 Keys, einmal klassisch und einmal PQC

-Vorteil: gute Übergang von klassische auf PCQ-Signaturen durch Abwärtskompatibilität + Sicherheit

Quellen

https://lists.gnupg.org/pipermail/gnutls-help/2024-November/004865.html
https://www.namecheap.com/blog/beginners-guide-to-tls-cipher-suites/
https://lists.gnupg.org/pipermail/gnutls-help/2025-February/004875.html

https://lists.gnupg.org/pipermail/gnutls-help/2025-July/004883.html
https://datatracker.ietf.org/doc/draft-ietf-tls-ecdhe-mlkem/
https://www.wolfssl.com/products/wolfcrypt-post-quantum/
https://nvlpubs.nist.gov/nistpubs/fips/nist.fips.203.pdf#:~:text=The%20computational%20assumption,KEM
https://learnmeabitcoin.com/technical/cryptography/elliptic-curve/
https://thibautprobst.fr/en/posts/ml-kem/
https://docs.securosys.com/tsb/Tutorials/Post-Quantum-Cryptography/key-encapsulation-mechanism/
https://www.digicert.com/tls-ssl/post-quantum-cryptography
https://docs.keyfactor.com/ejbca/9.0/hybrid-ca

https://www.sectigo.com/resource-library/what-are-quantum-safe-and-hybrid-certificates