OpenSSL Conference
Prague 2025

IBM **Research** Security

In-Situ Performance Measurements of Crypto-Algorithms in TLS v1.3

Dr. Martin L. Schmatz

IBM Research -Zurich

Oct 09, 2025

IBM

Motivation

Clients are concerned about the ***processing efforts*** when using ***Q-safe TLS*** connections for:

- Key establishment
- Authentication (~= certificates)
- Confidentiality & Authenticity (~=enc-/de-cryption)

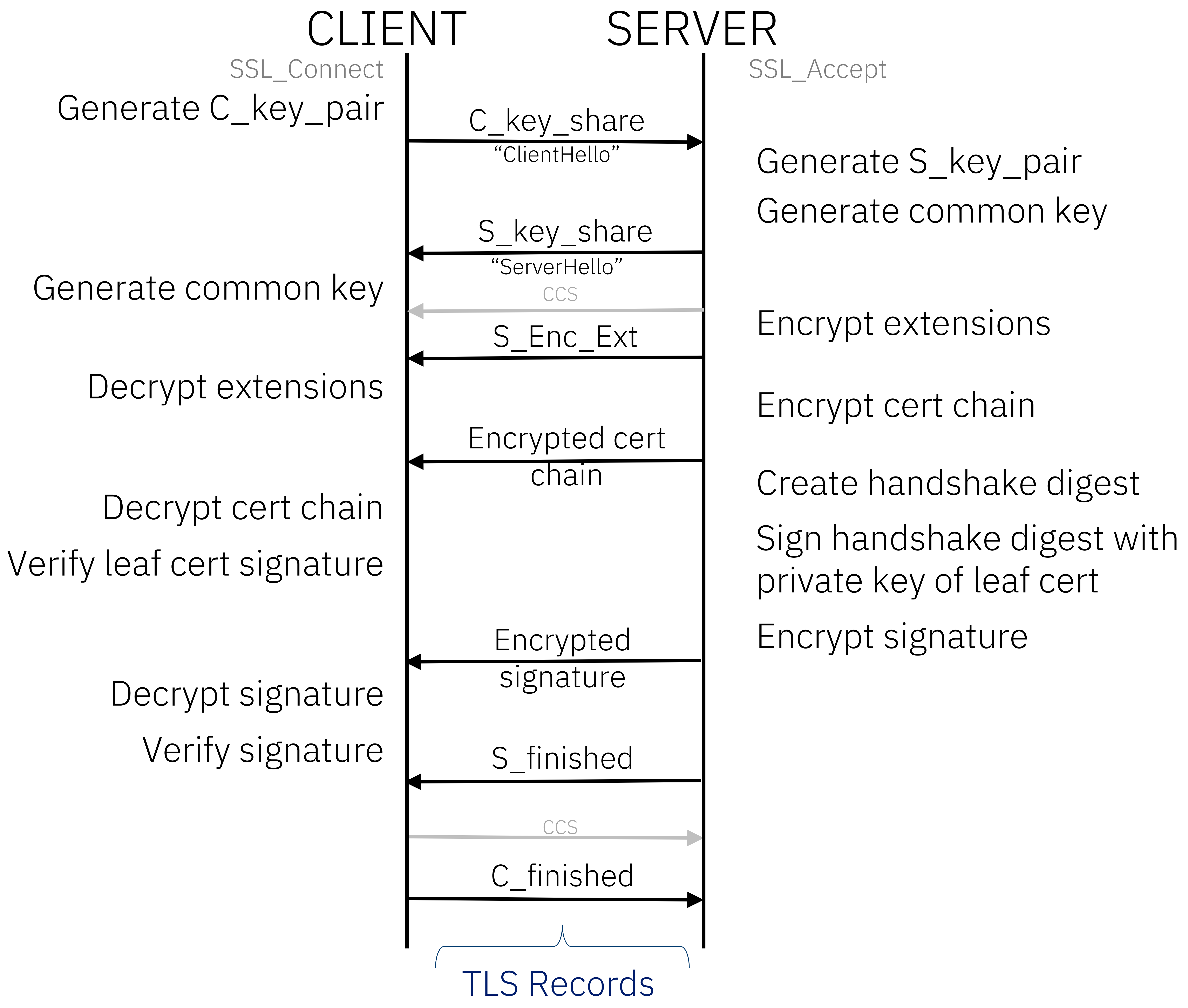
Clients are concerned about the ***required network bandwidth*** when using ***Q-safe TLS*** connections

- While synthetic (cycle counting) and measured performance for the processing of Q-safe algorithms are readily available, those DO NOT include *collateral performance penalties* for the *use of the algorithms in the TLS v1.3* ***protocol*** into account (e.g., memory allocations, data copying, algorithm loading, merging hybrid secrets, etc.)

- Hence the desire to get ***in-situ performance measurements:***
CPU time required to execute the individual steps of a TLS protocol session establishment and the *data volume* to be transported over the network *as a function of Q-safe algorithms*.

Short Intro: TLS v1.3 Session Establishment

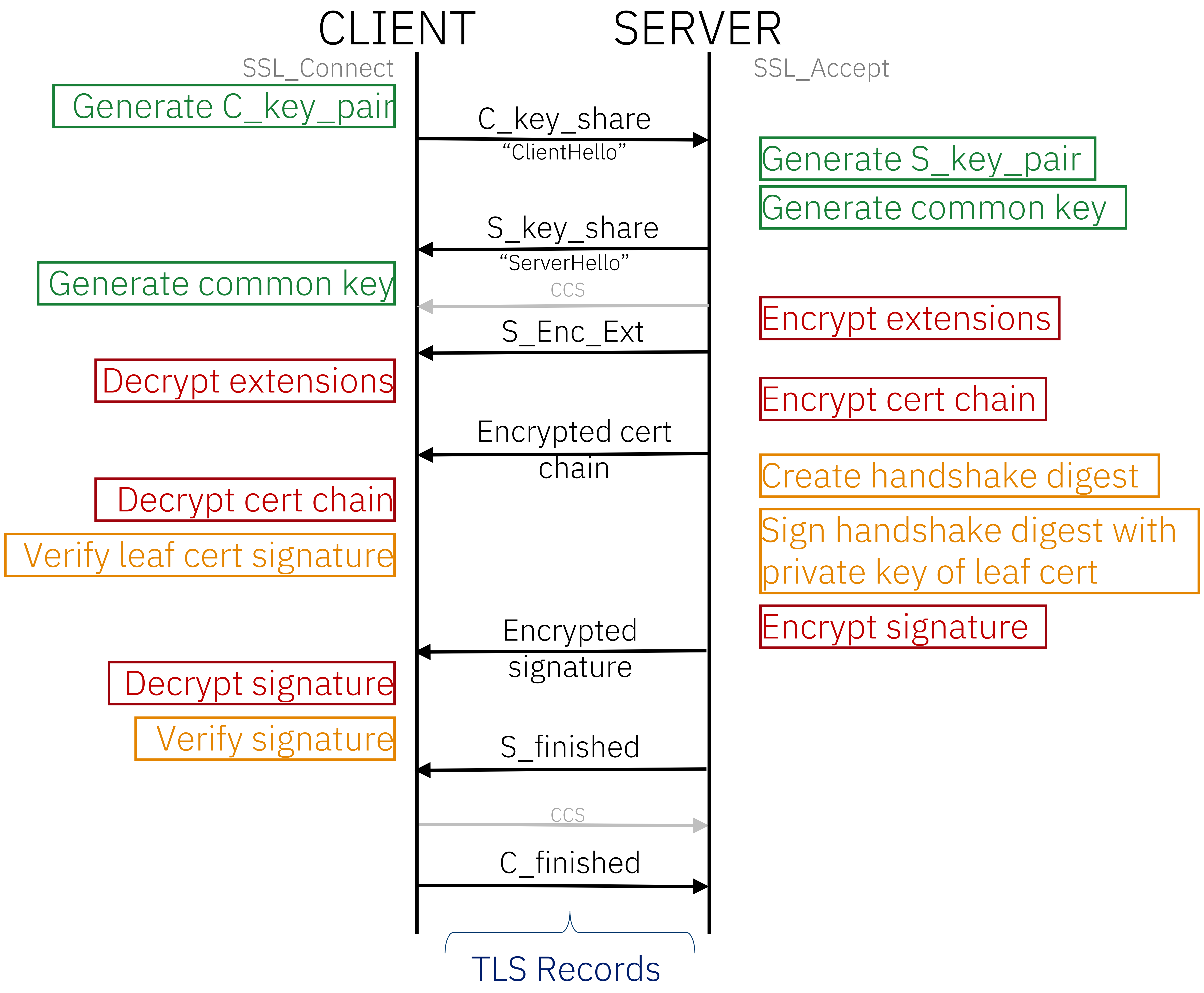
Only relevant parts shown, no HelloRetryRequest (key share fits supported groups)



Short Intro: TLS v1.3 Session Establishment

Only relevant parts shown, no HelloRetryRequest (key share fits supported groups)

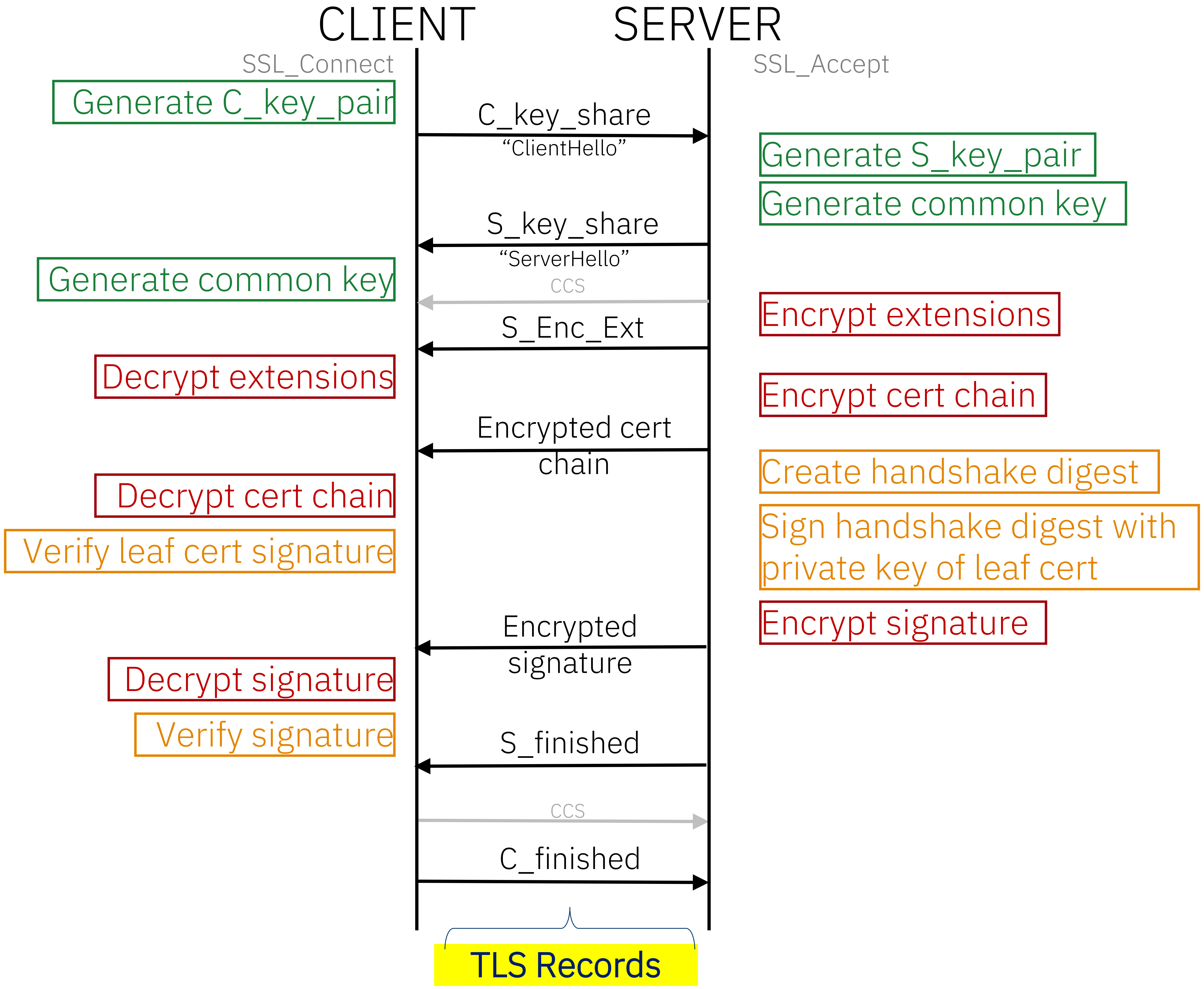
Some processing is for key establishment, for confidentiality & authenticity, and for authentication



Short Intro: TLS v1.3 Session Establishment

Only relevant parts shown, no HelloRetryRequest (key share fits supported groups)

Some processing is for key establishment, for confidentiality & authenticity, and for authentication



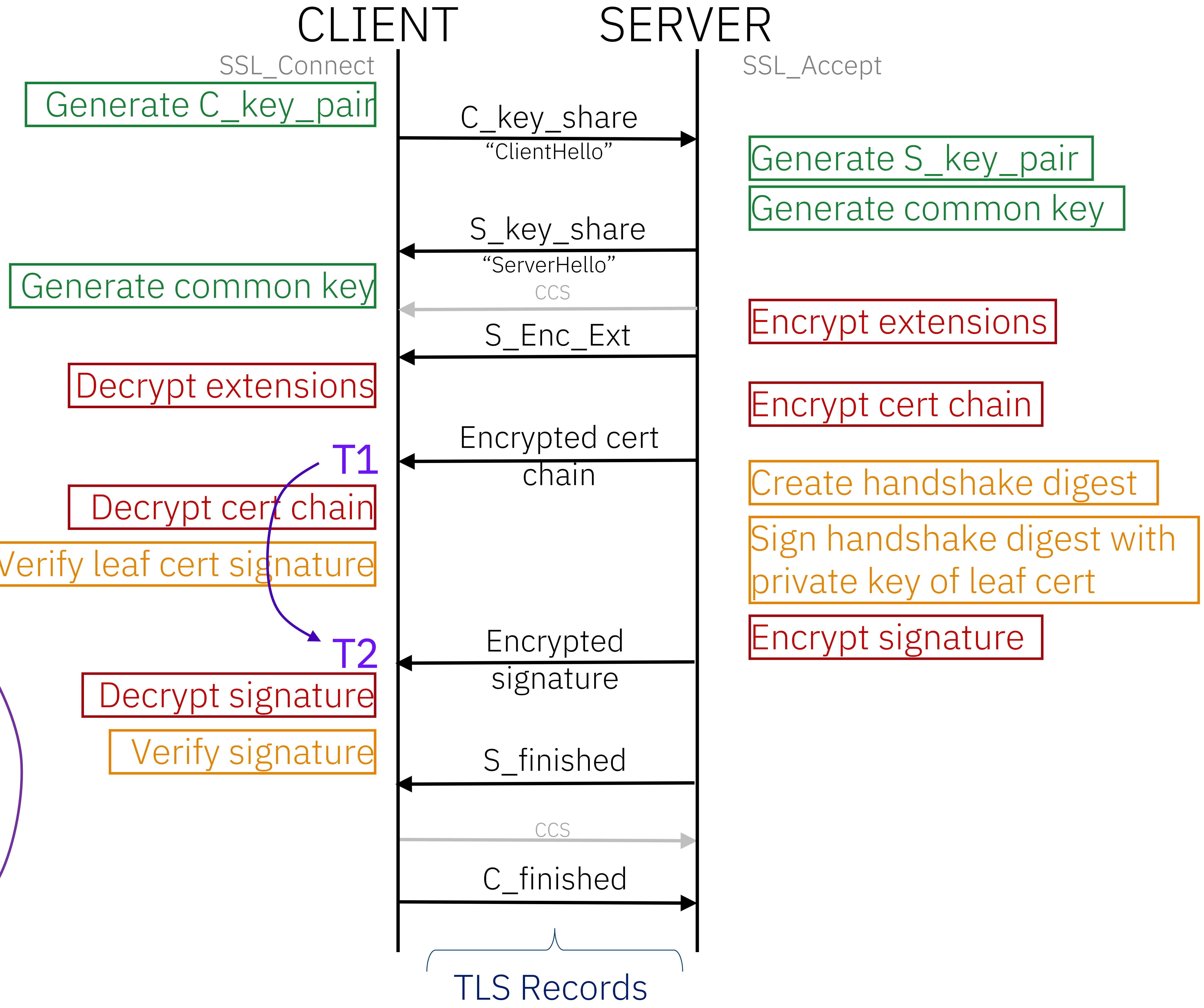
Short Intro: TLS v1.3 Session Establishment

Only relevant parts shown, no HelloRetryRequest (key share fits supported groups)

Some processing is for key establishment, for confidentiality & authenticity, and for authentication

The measured timestamps of TLS record *processing* start/end can be used to get CPU time used for the computations

Example:
T2-T1 = Time to decrypt cert chain and to verify cert signature(s)



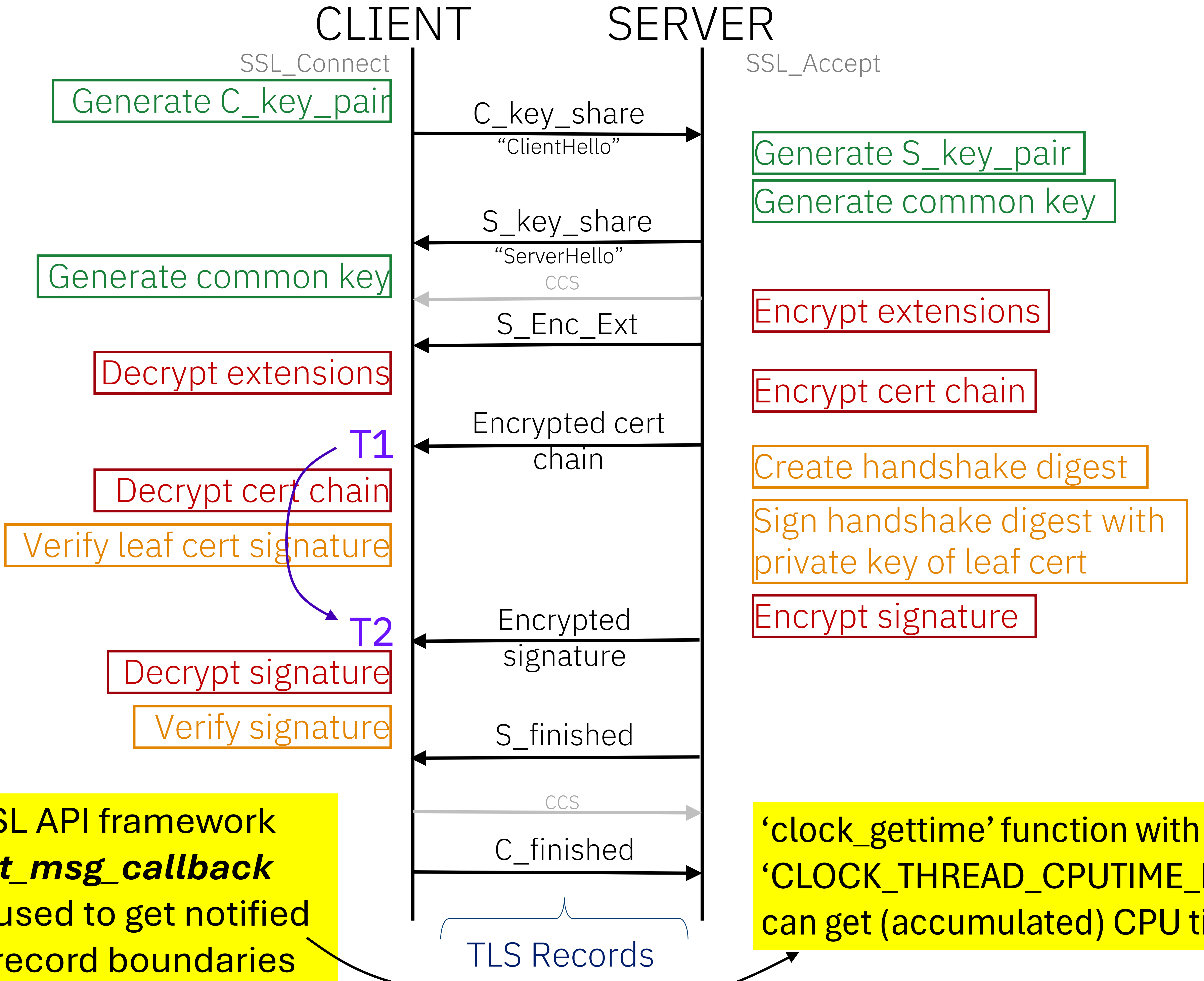
Short Intro: TLS v1.3 Session Establishment

Only relevant parts shown, no HelloRetryRequest (key share fits supported groups)

Some processing is for key establishment, for confidentiality & authenticity, and for authentication

The measured timestamps of TLS record *processing* start/end can be used to get CPU time used for the computations

Example:
T2-T1 = Time to decrypt cert chain and to verify cert signature

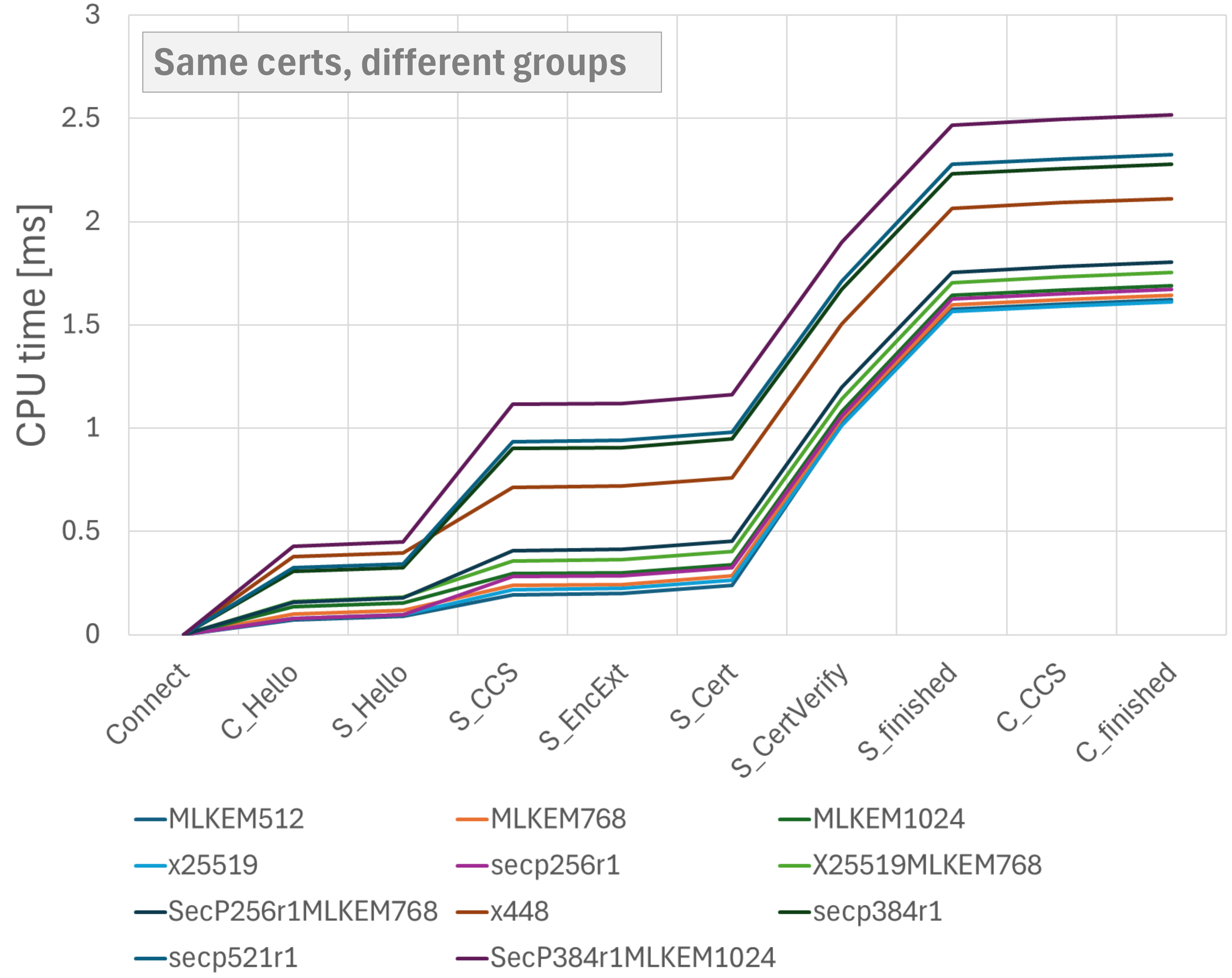


OpenSSL API framework *SSL_set_msg_callback* can be used to get notified on TLS record boundaries

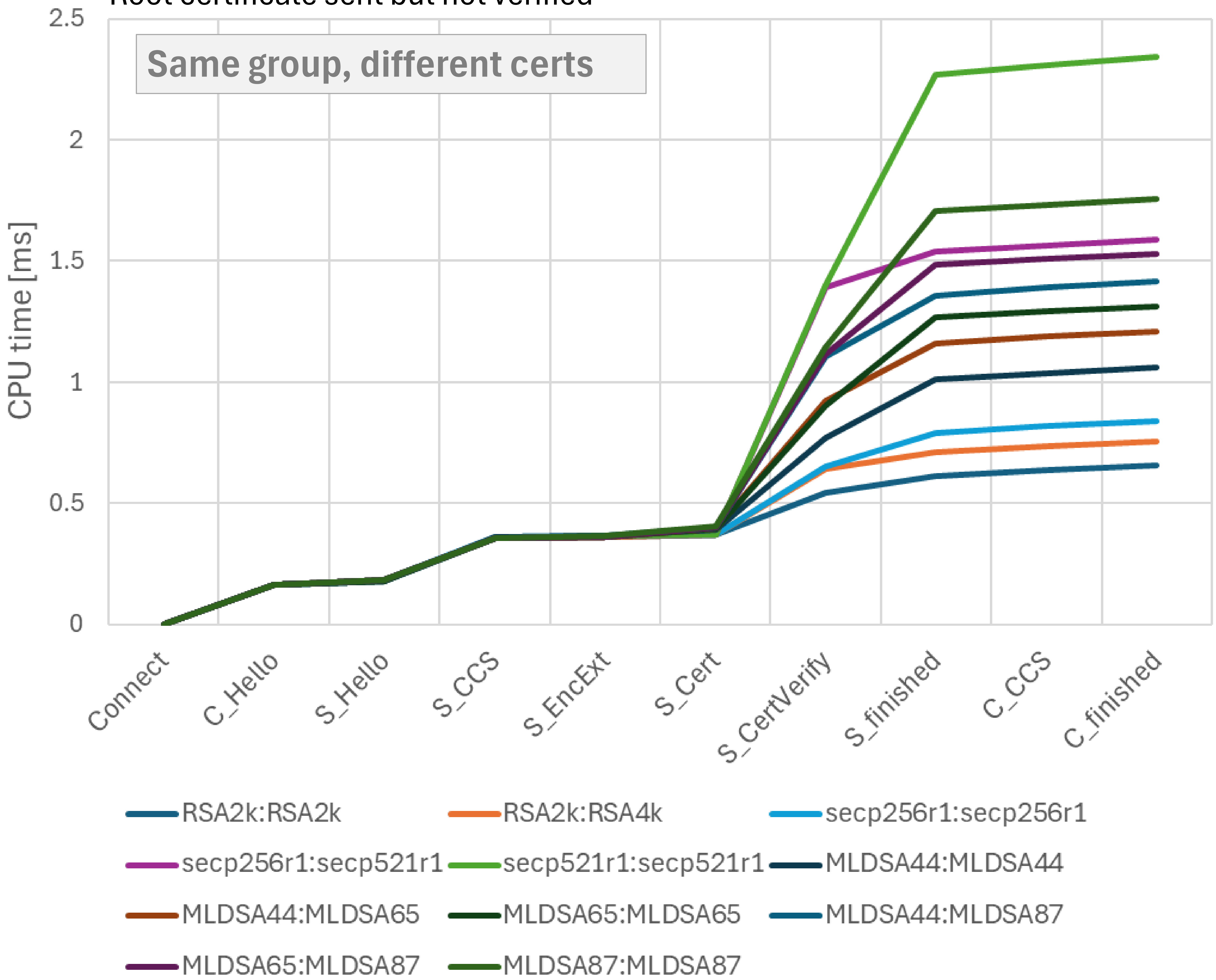
'clock_gettime' function with 'CLOCK_THREAD_CPUTIME_ID' can get (accumulated) CPU time

Measurement examples: CLIENT

Constant:
2-stage cert chain MLDSA87:MLDSA87
TLS_AES_128_GCM_SHA256
Root certificate sent but not verified



Constant:
X25519MLKEM768
TLS_AES_128_GCM_SHA256
Root certificate sent but not verified

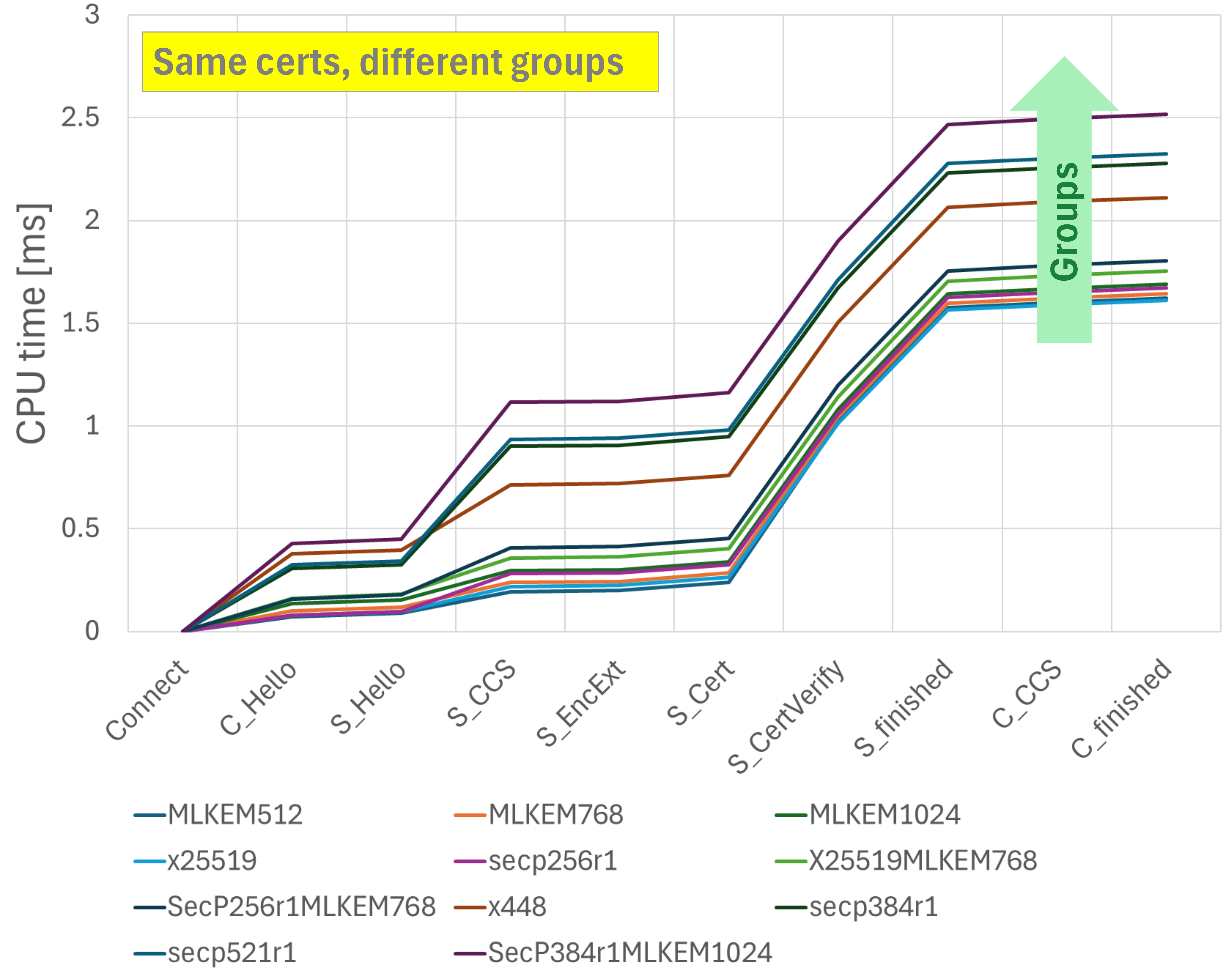


Certificate notation: <Leaf cert>:<Root cert>

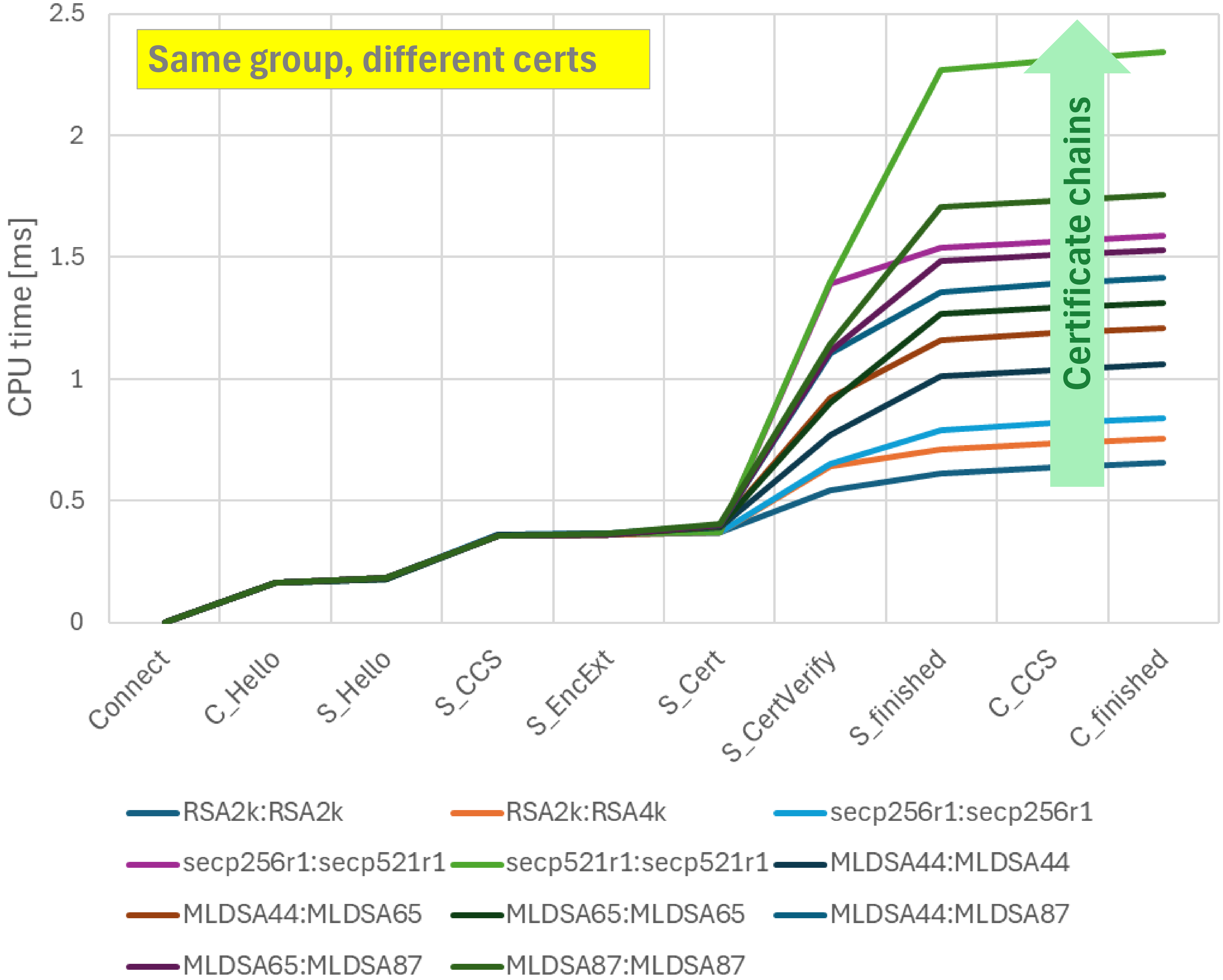
Note: For illustrative purposes only, generated with OpenSSL v3.5.0

Measurement examples: CLIENT

Constant:
2-stage cert chain MLDSA87:MLDSA87
TLS_AES_128_GCM_SHA256
Root certificate sent but not verified



Constant:
X25519MLKEM768
TLS_AES_128_GCM_SHA256
Root certificate sent but not verified

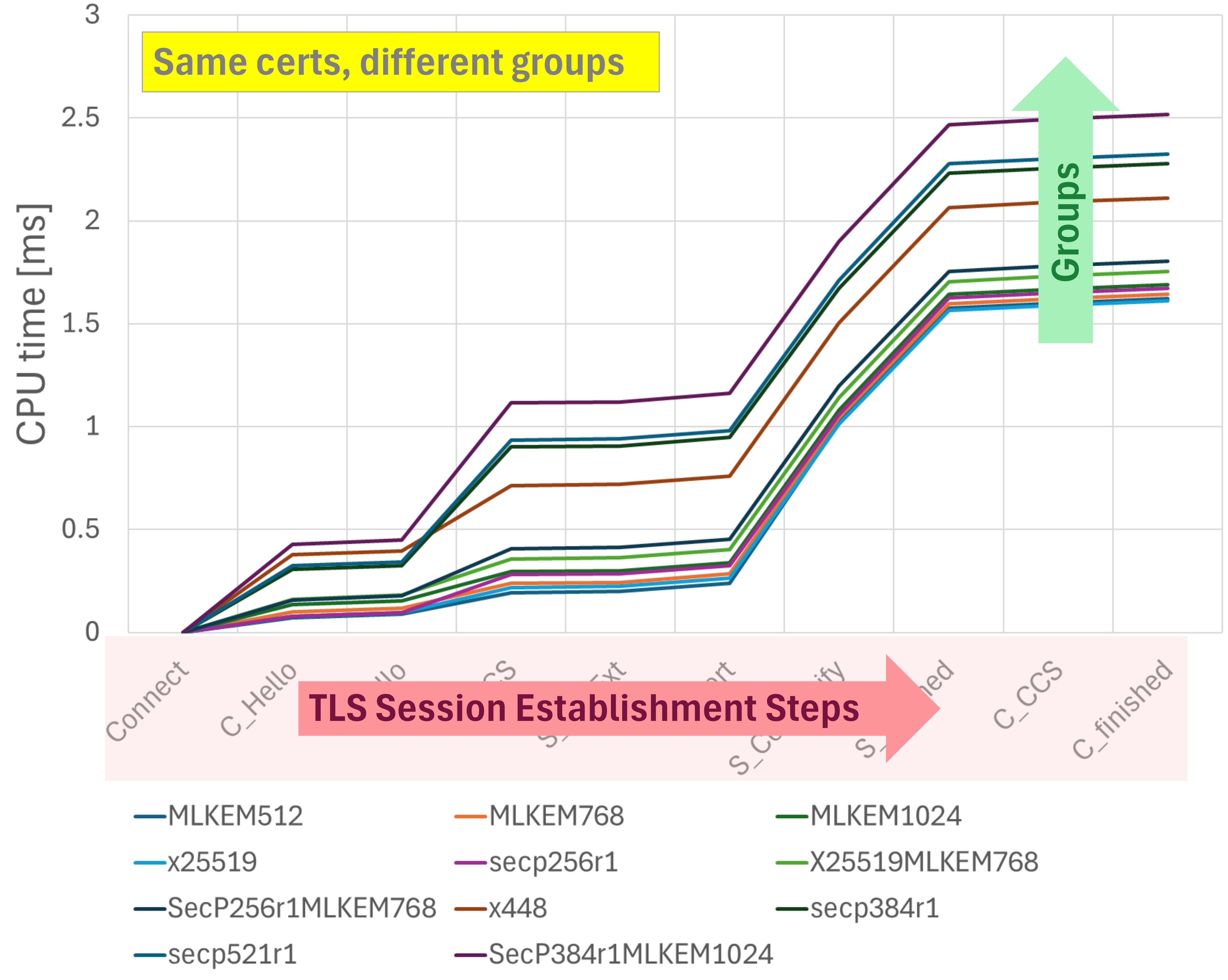


Certificate notation: <Leaf cert>:<Root cert>

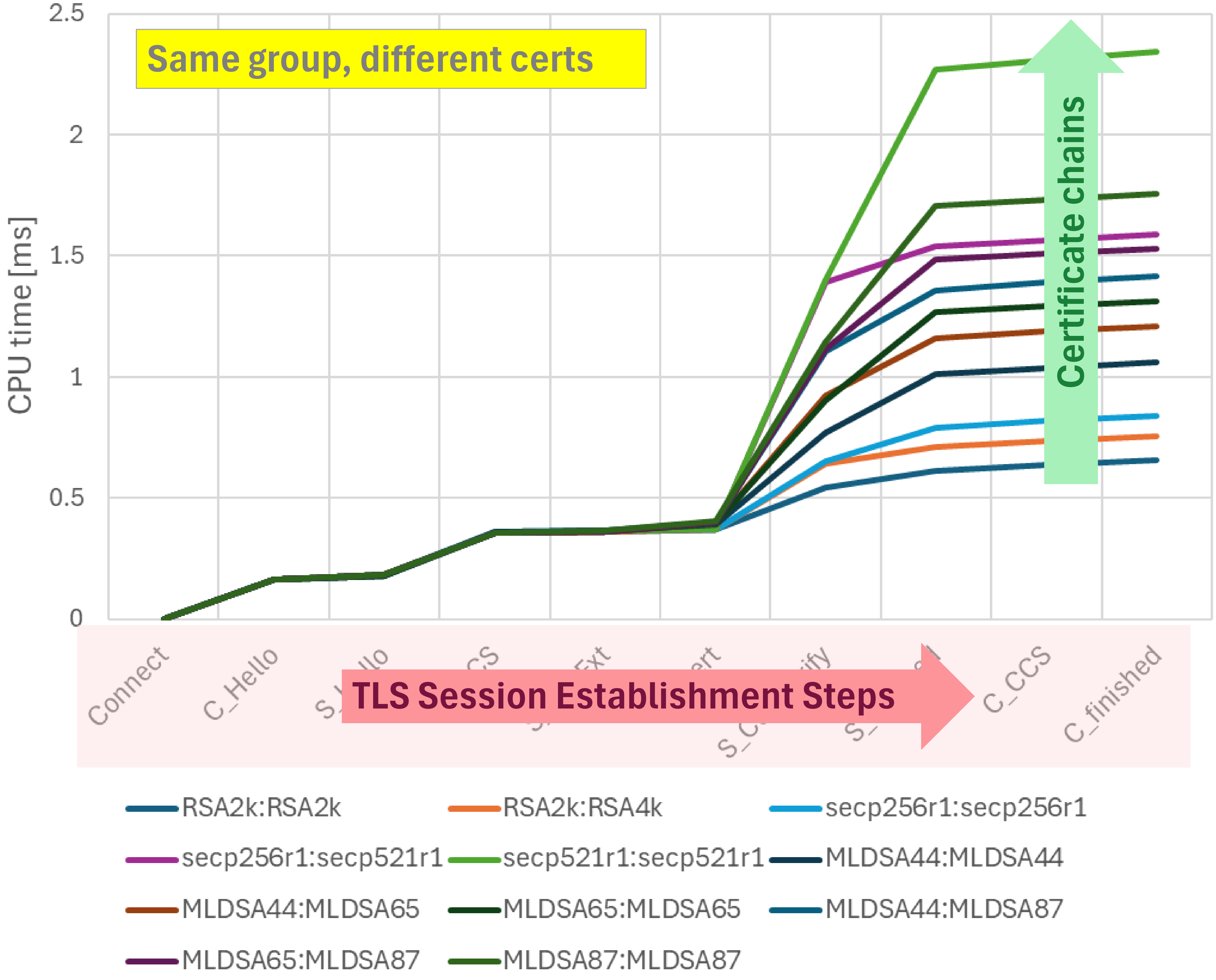
Note: For illustrative purposes only, generated with OpenSSL v3.5.0

Measurement examples: CLIENT

Constant:
2-stage cert chain MLDSA87:MLDSA87
TLS_AES_128_GCM_SHA256
Root certificate sent but not verified



Constant:
X25519MLKEM768
TLS_AES_128_GCM_SHA256
Root certificate sent but not verified

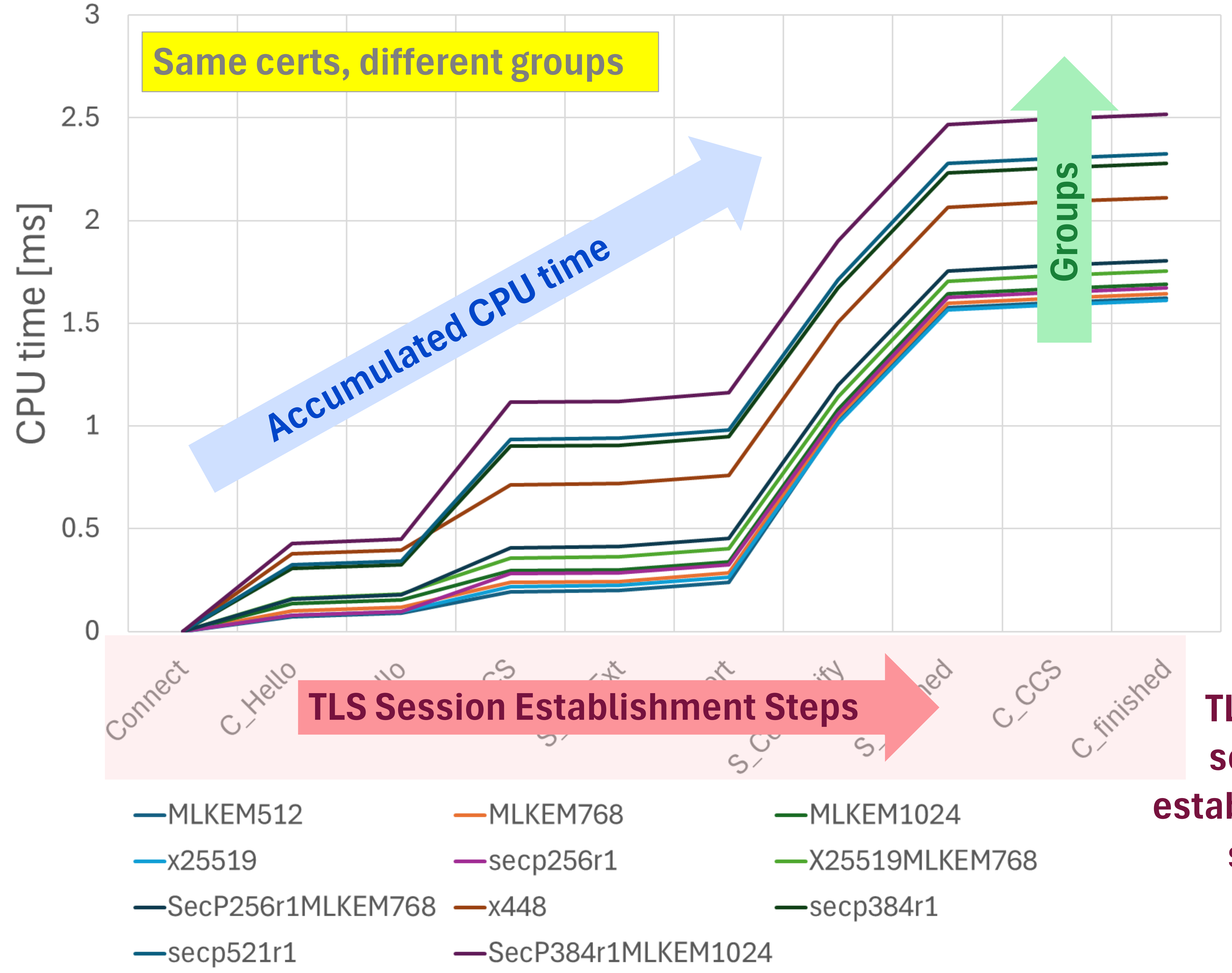


Certificate notation: <Leaf cert>:<Root cert>

Note: For illustrative purposes only, generated with OpenSSL v3.5.0

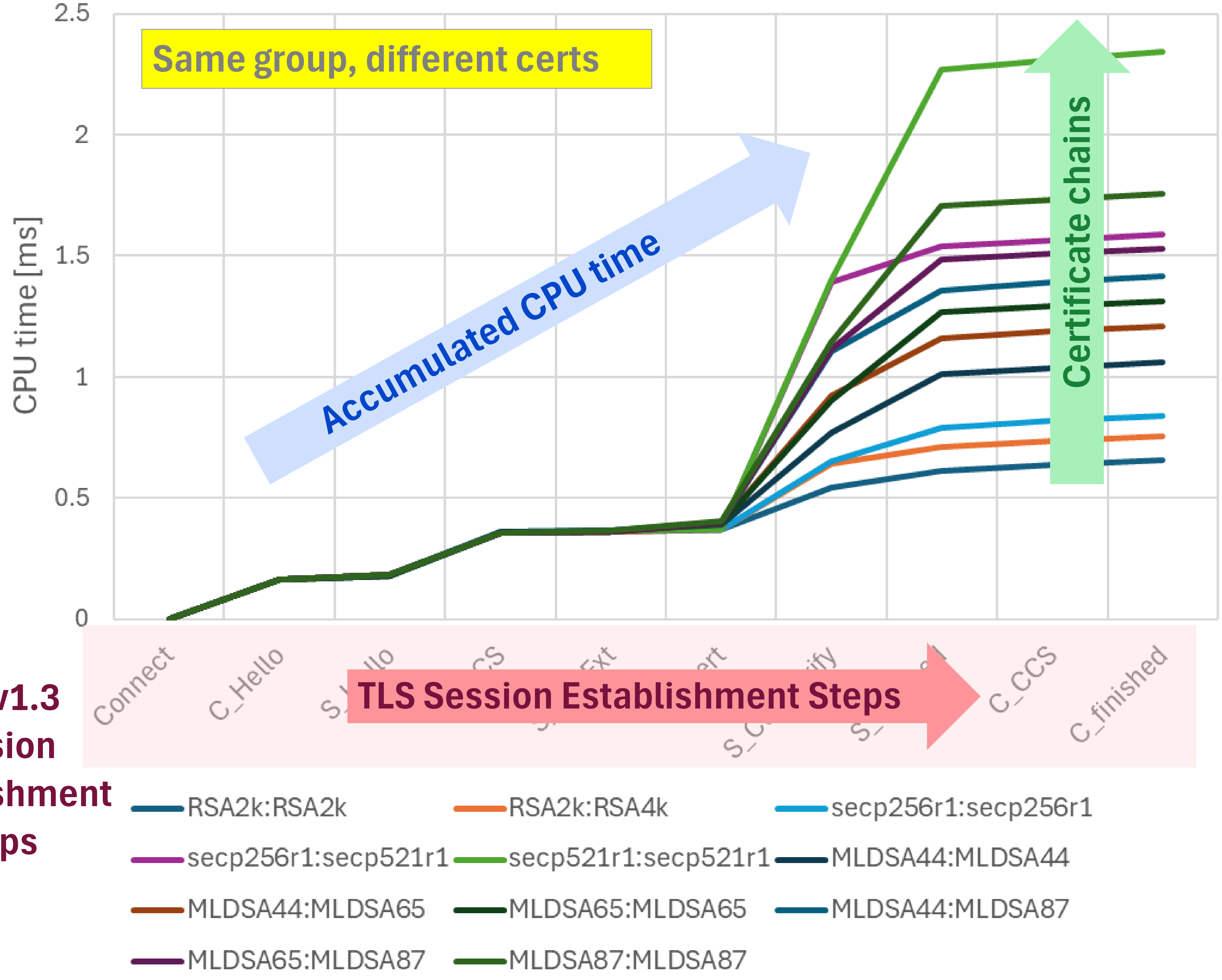
Measurement examples: CLIENT

Constant:
2-stage cert chain MLDSA87:MLDSA87
TLS_AES_128_GCM_SHA256
Root certificate sent but not verified



TLS v1.3 session establishment steps

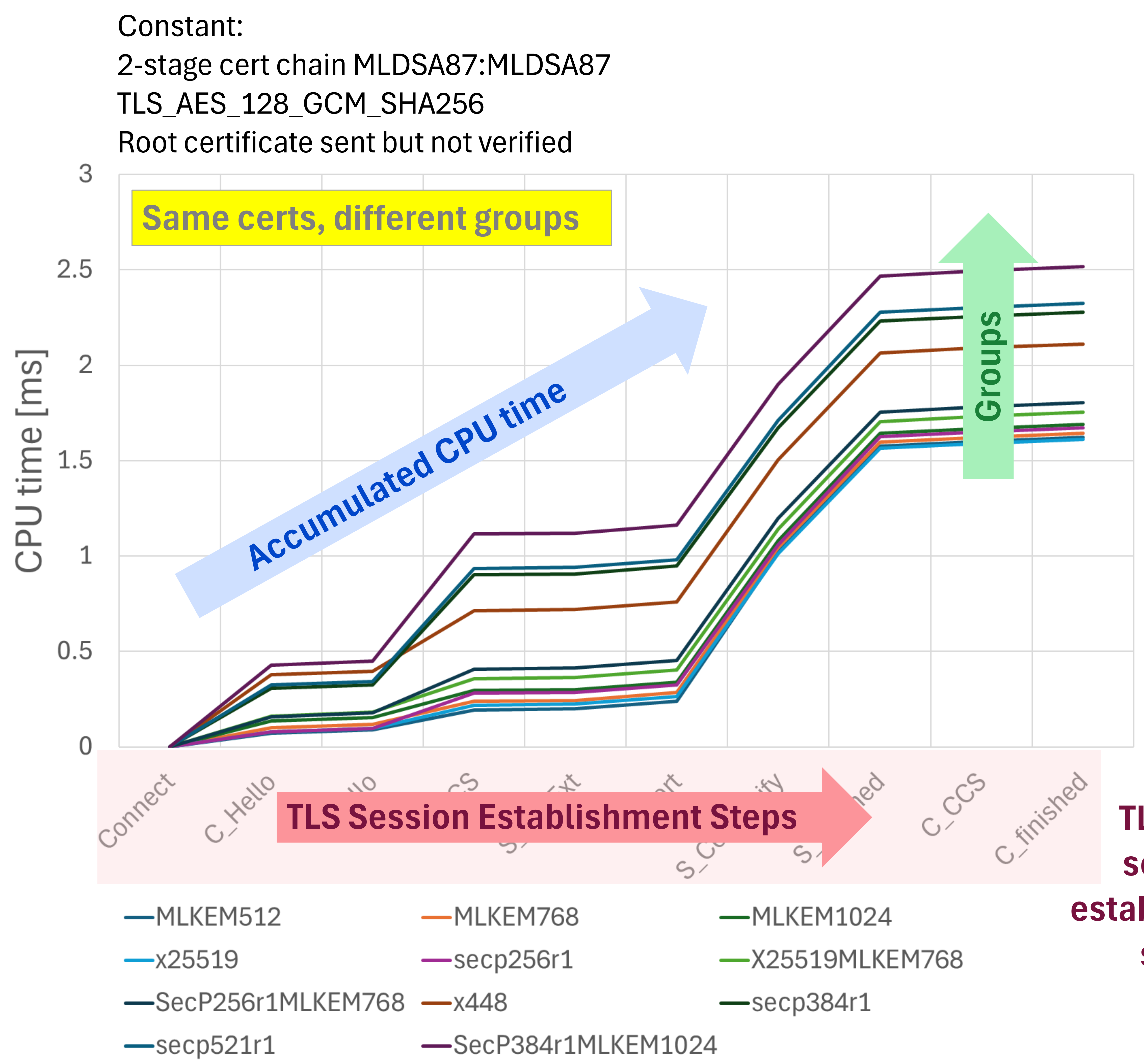
Constant:
X25519MLKEM768
TLS_AES_128_GCM_SHA256
Root certificate sent but not verified



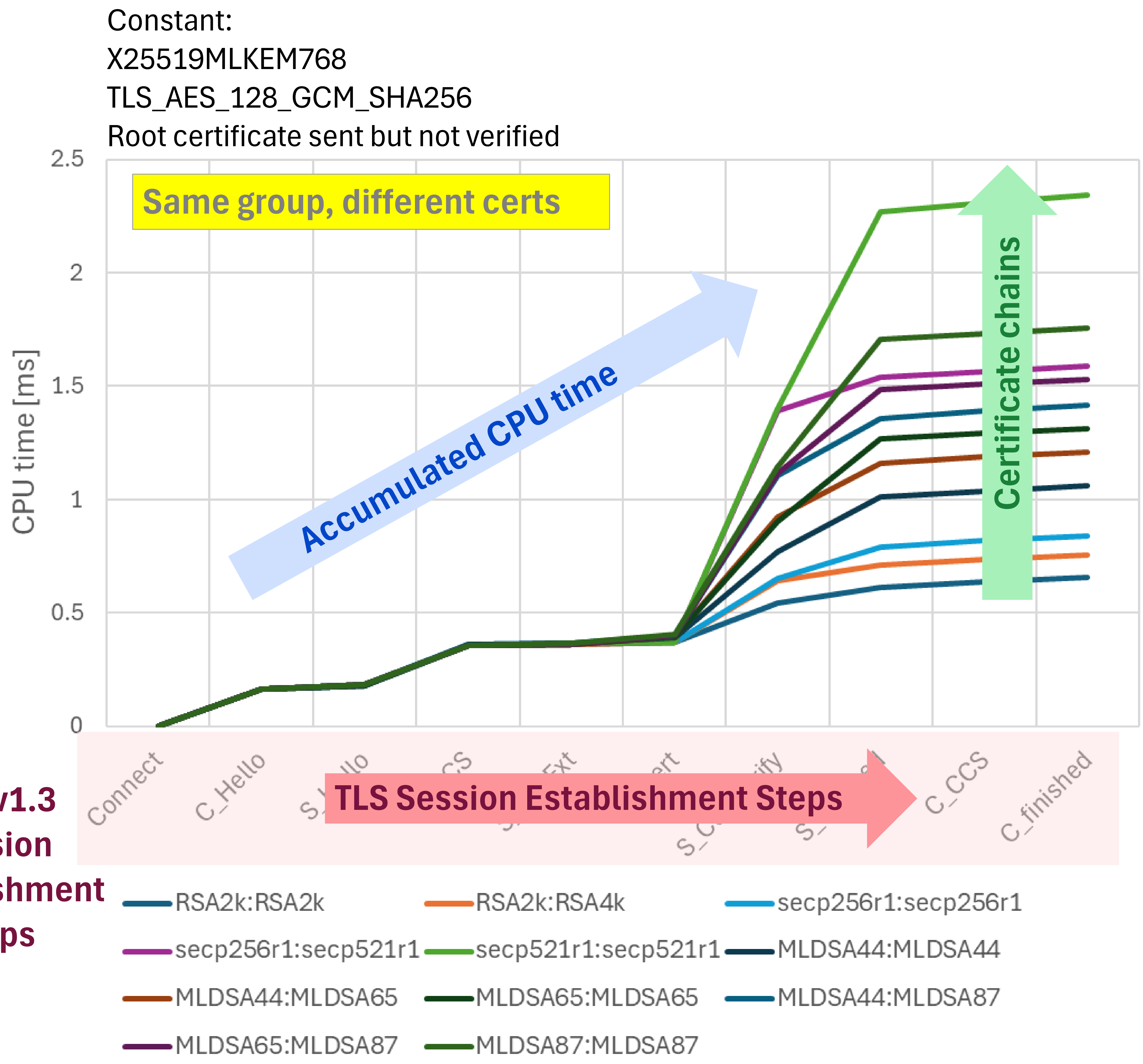
Certificate notation: <Leaf cert>:<Root cert>

Note: For illustrative purposes only, generated with OpenSSL v3.5.0

Measurement examples: CLIENT



TLS v1.3 session establishment steps



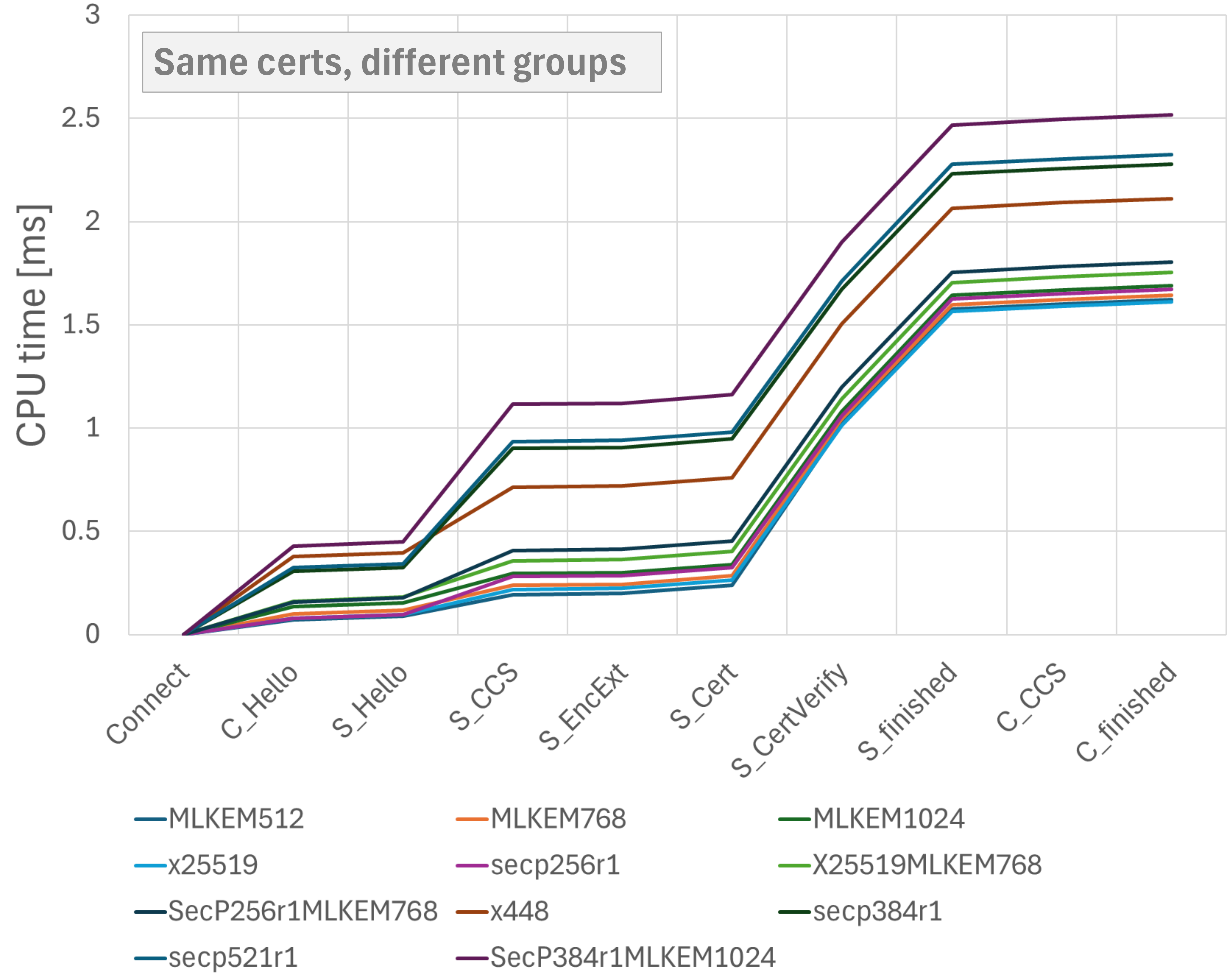
Certificate notation: <Leaf cert>:<Root cert>

Key establishment and certificate processing are orthogonal

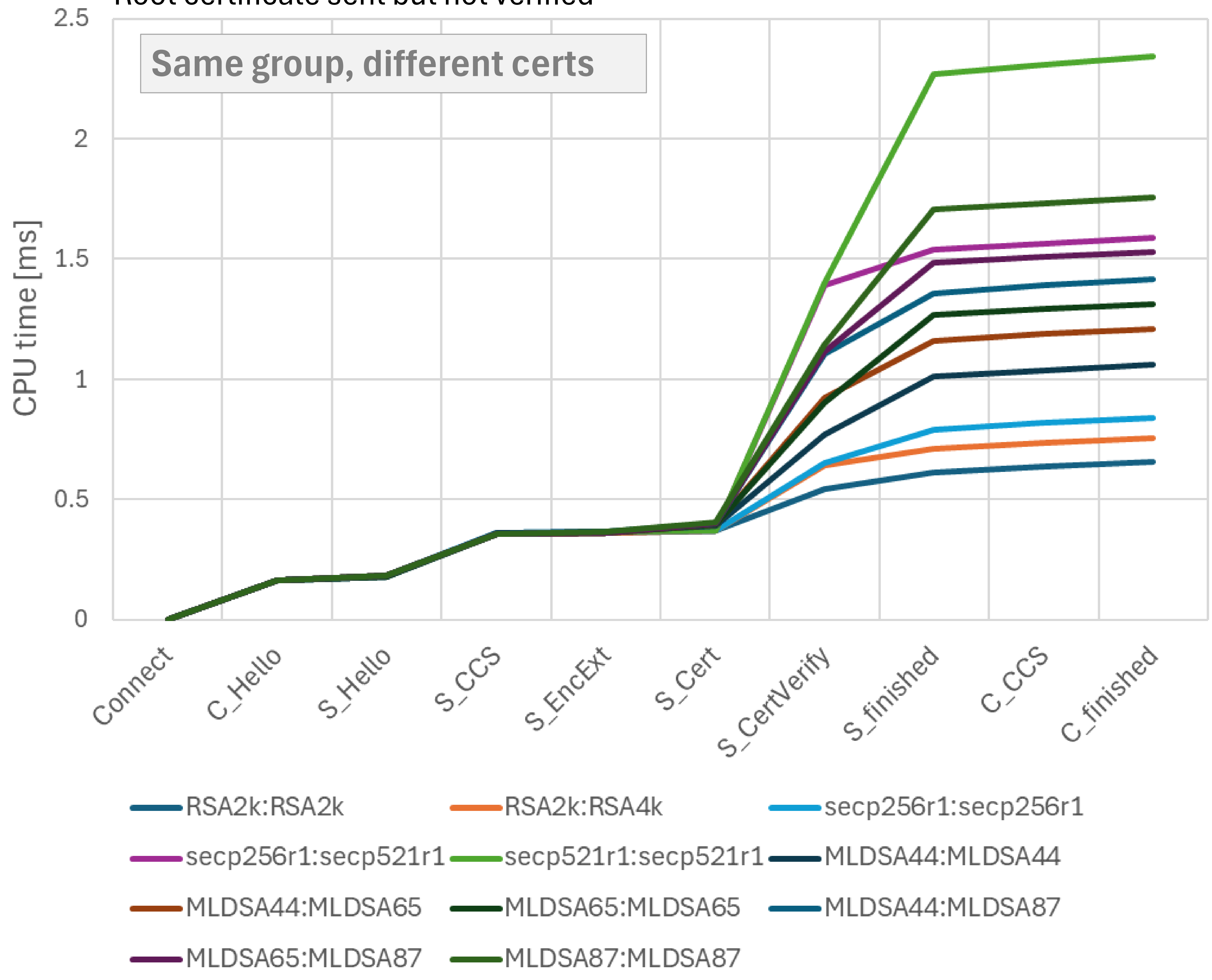
Note: For illustrative purposes only, generated with OpenSSL v3.5.0

Data extraction: CLIENT

Constant:
2-stage cert chain MLDSA87:MLDSA87
TLS_AES_128_GCM_SHA256
Root certificate sent but not verified



Constant:
X25519MLKEM768
TLS_AES_128_GCM_SHA256
Root certificate sent but not verified

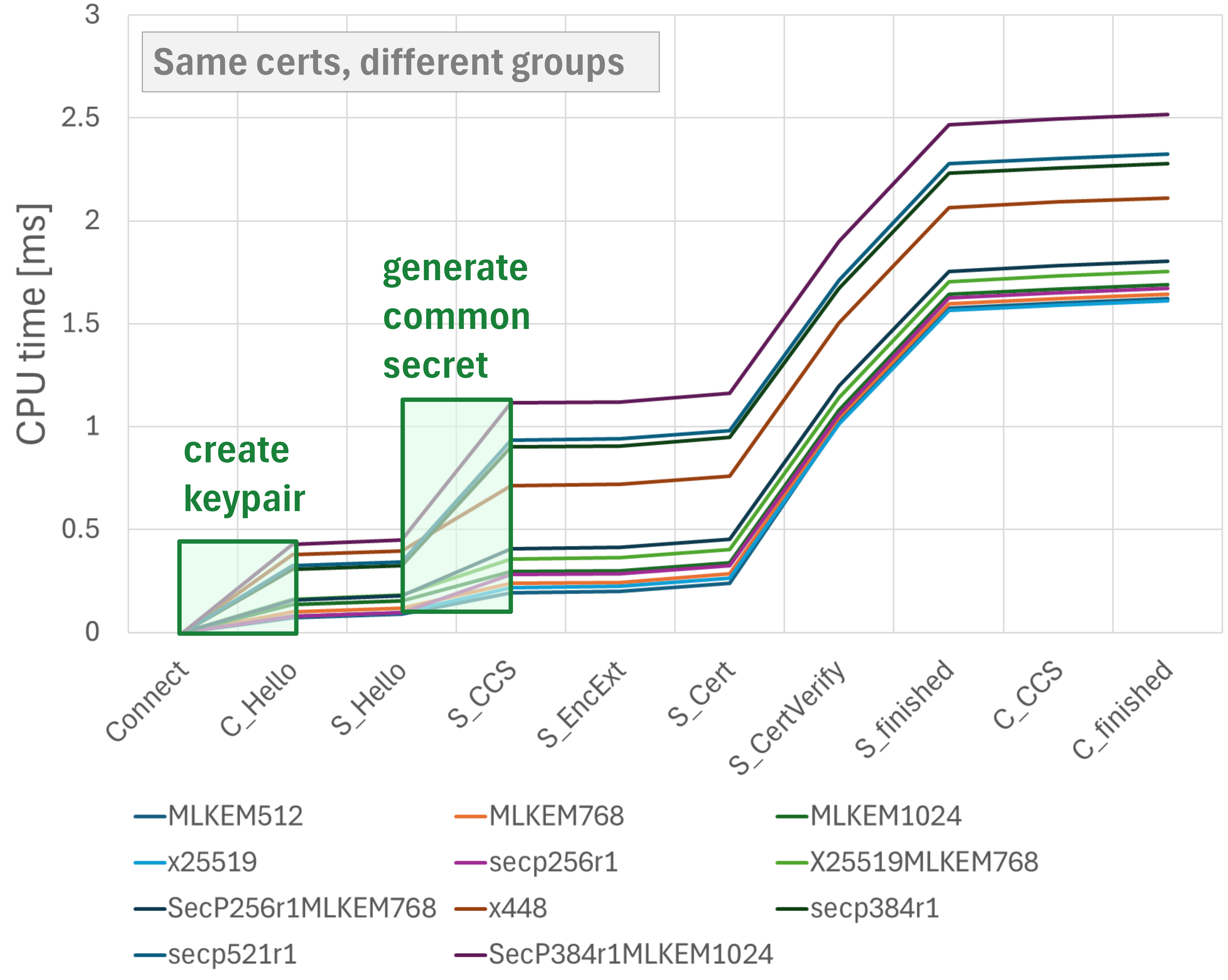


Certificate notation: <Leaf cert>:<Root cert>

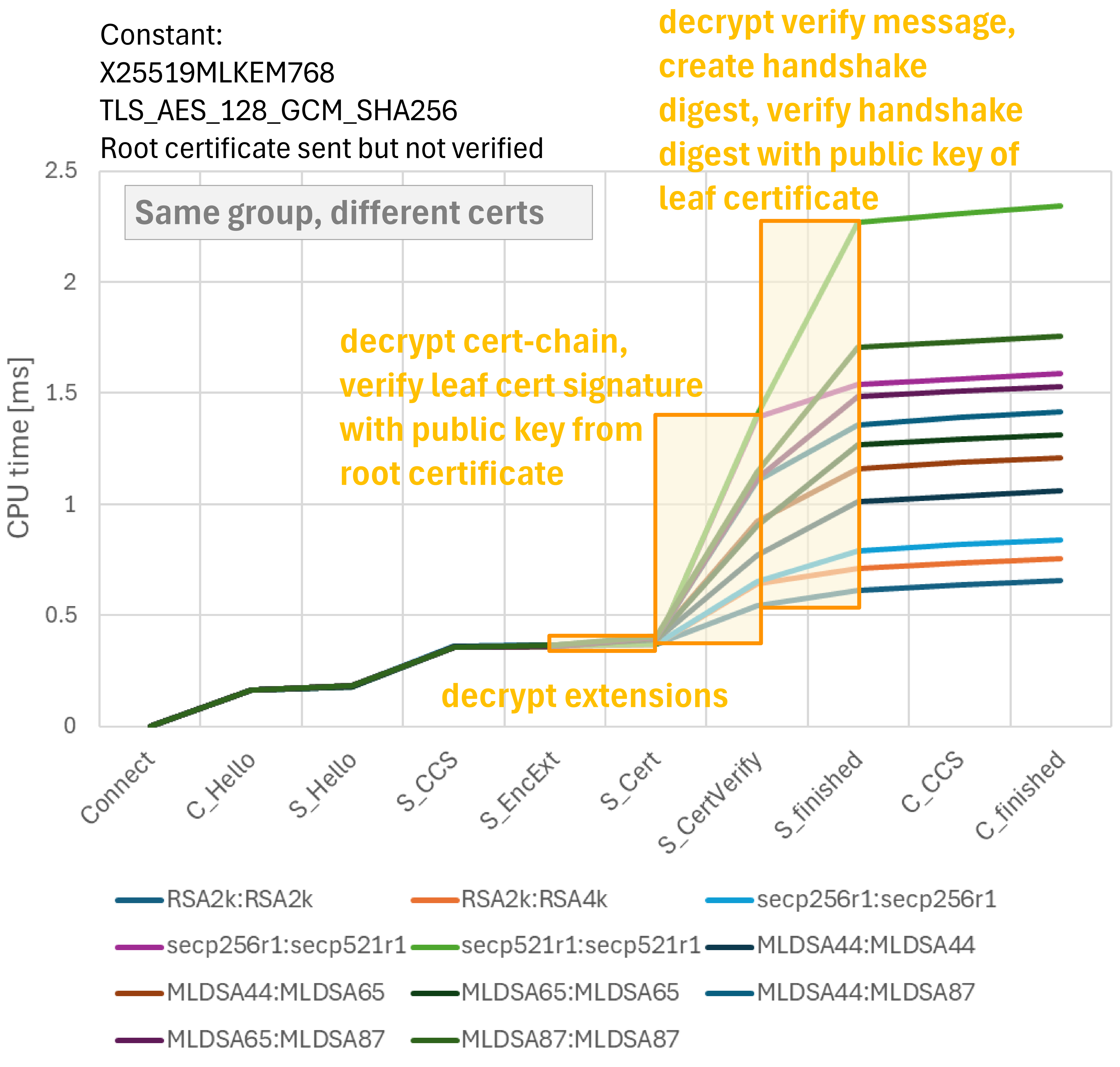
Note: For illustrative purposes only, generated with OpenSSL v3.5.0

Data extraction: CLIENT

Constant:
2-stage cert chain MLDSA87:MLDSA87
TLS_AES_128_GCM_SHA256
Root certificate sent but not verified



Constant:
X25519MLKEM768
TLS_AES_128_GCM_SHA256
Root certificate sent but not verified

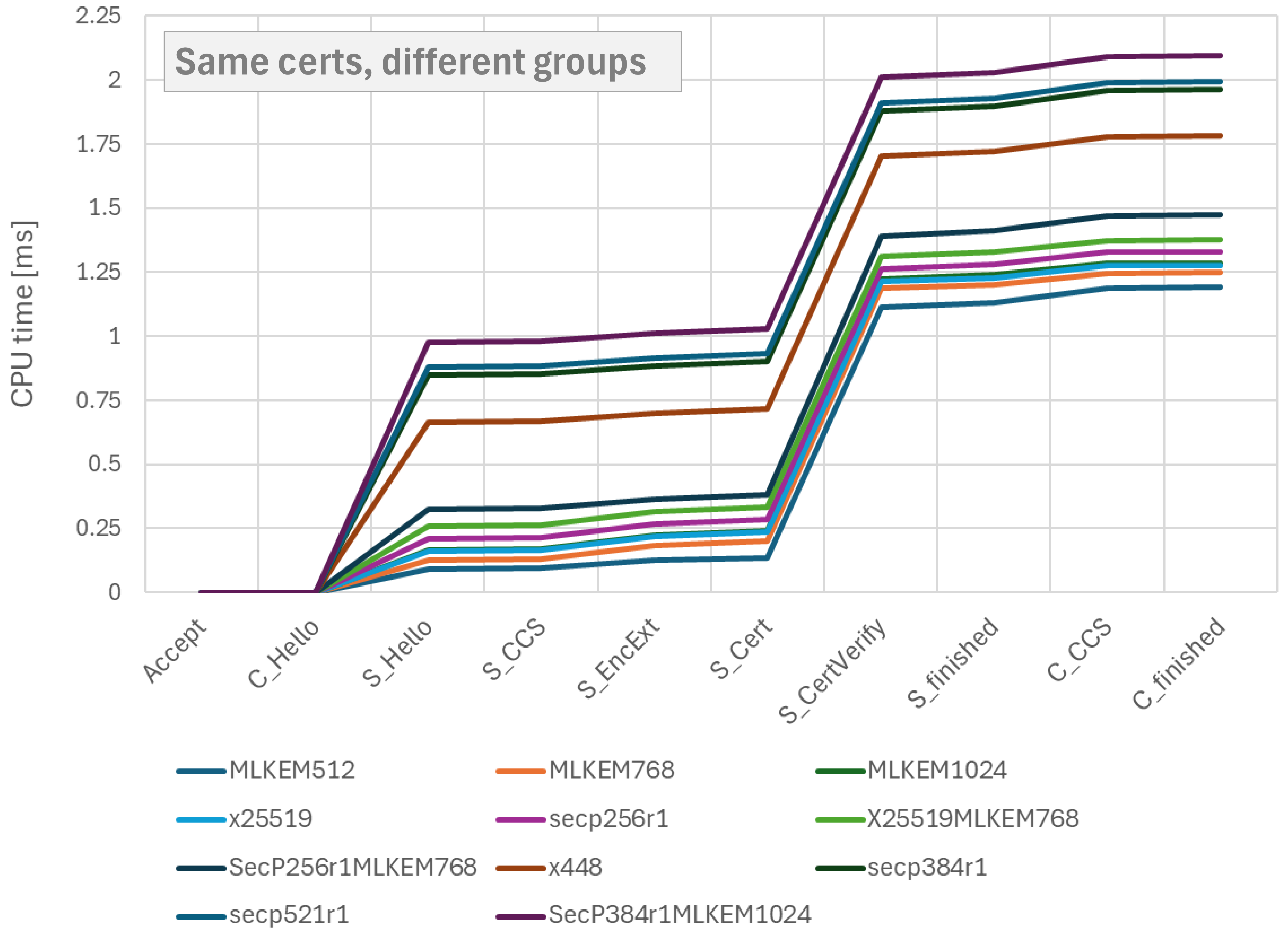


Certificate notation: <Leaf cert>:<Root cert>

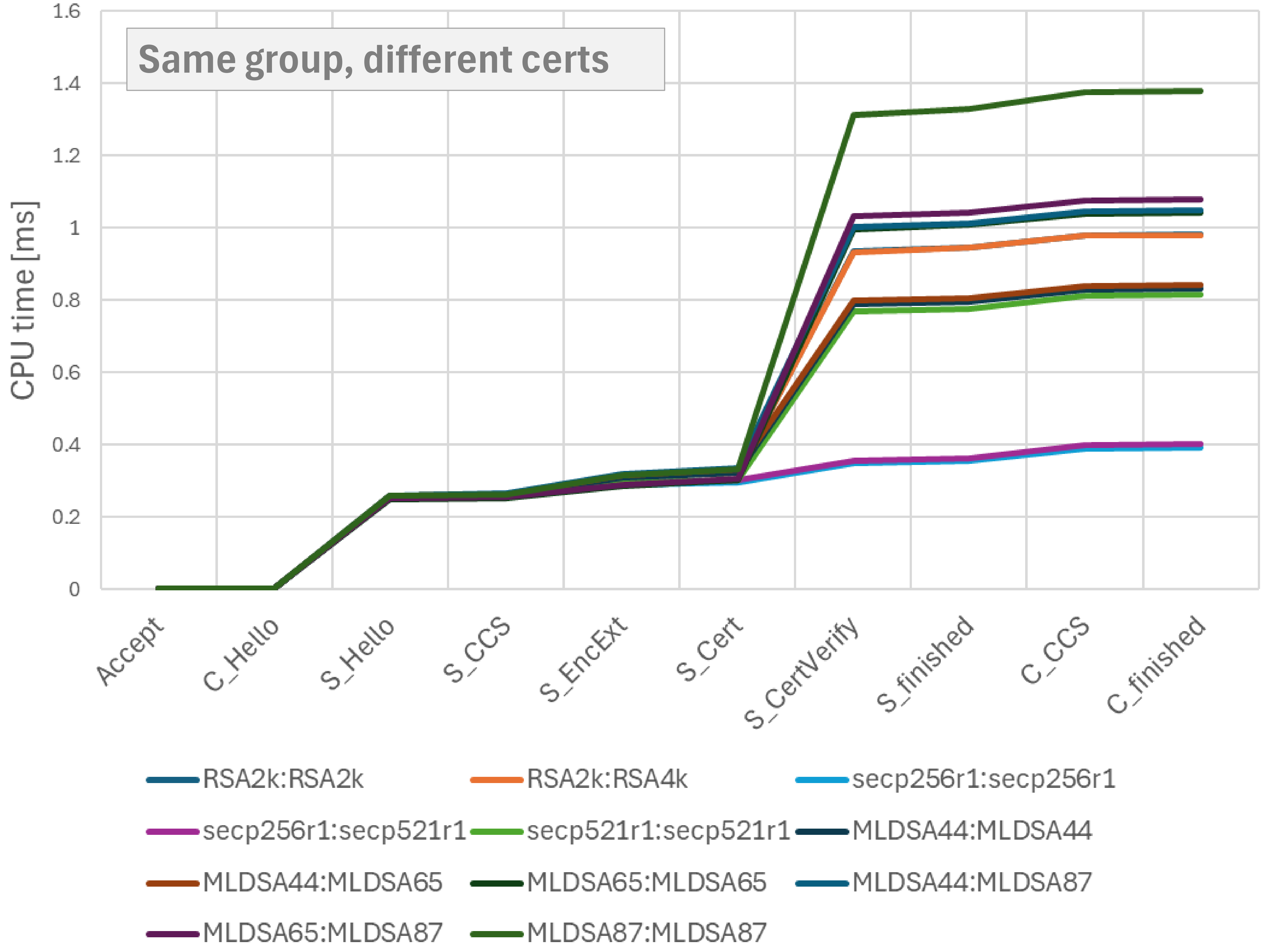
Note: For illustrative purposes only, generated with OpenSSL v3.5.0

Data extraction: SERVER

Constant:
2-stage cert chain MLDSA87:MLDSA87
TLS_AES_128_GCM_SHA256
Root certificate sent but not verified



Constant:
X25519MLKEM768
TLS_AES_128_GCM_SHA256
Root certificate sent but not verified

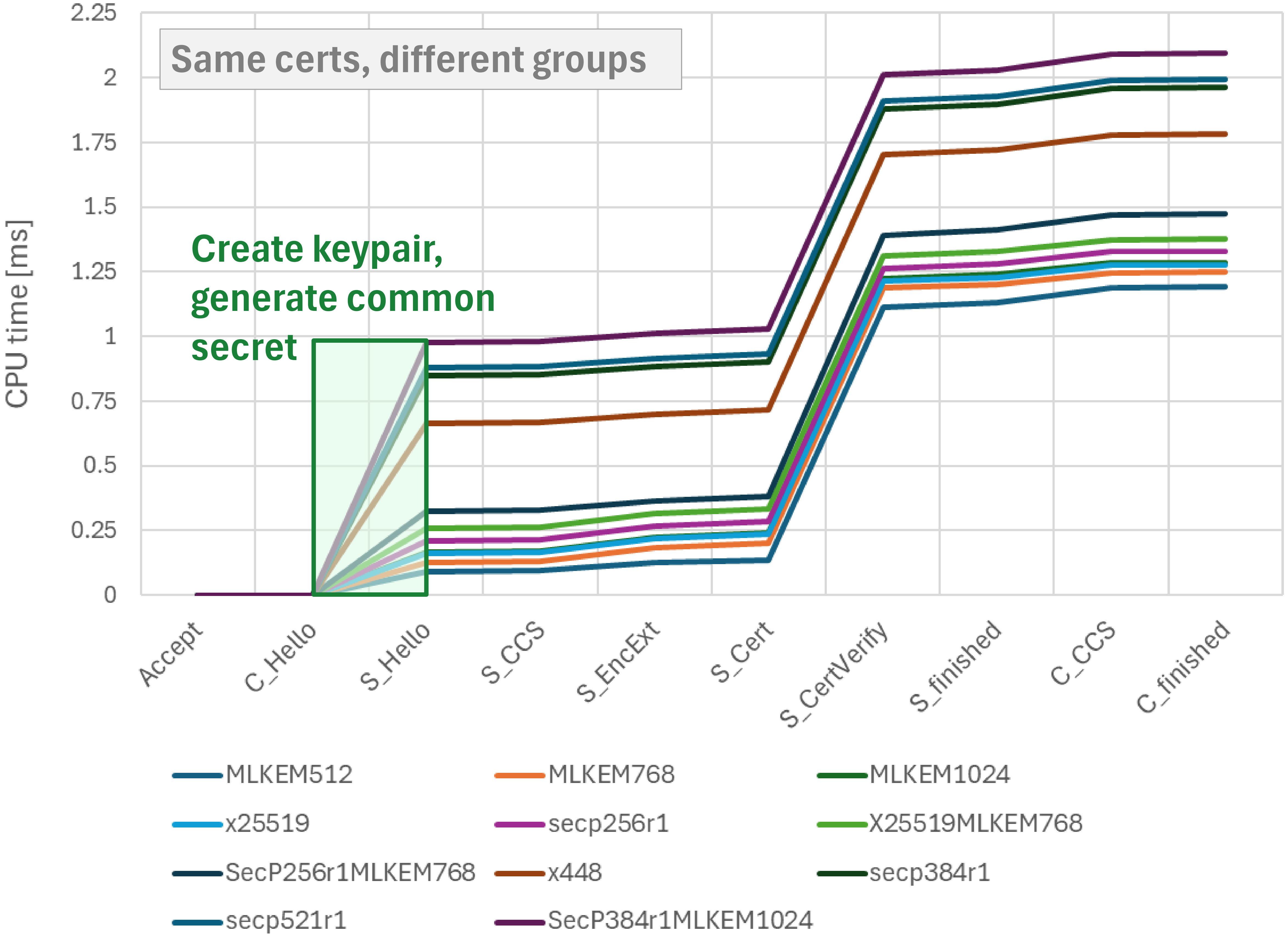


Certificate notation: <Leaf cert>:<Root cert>

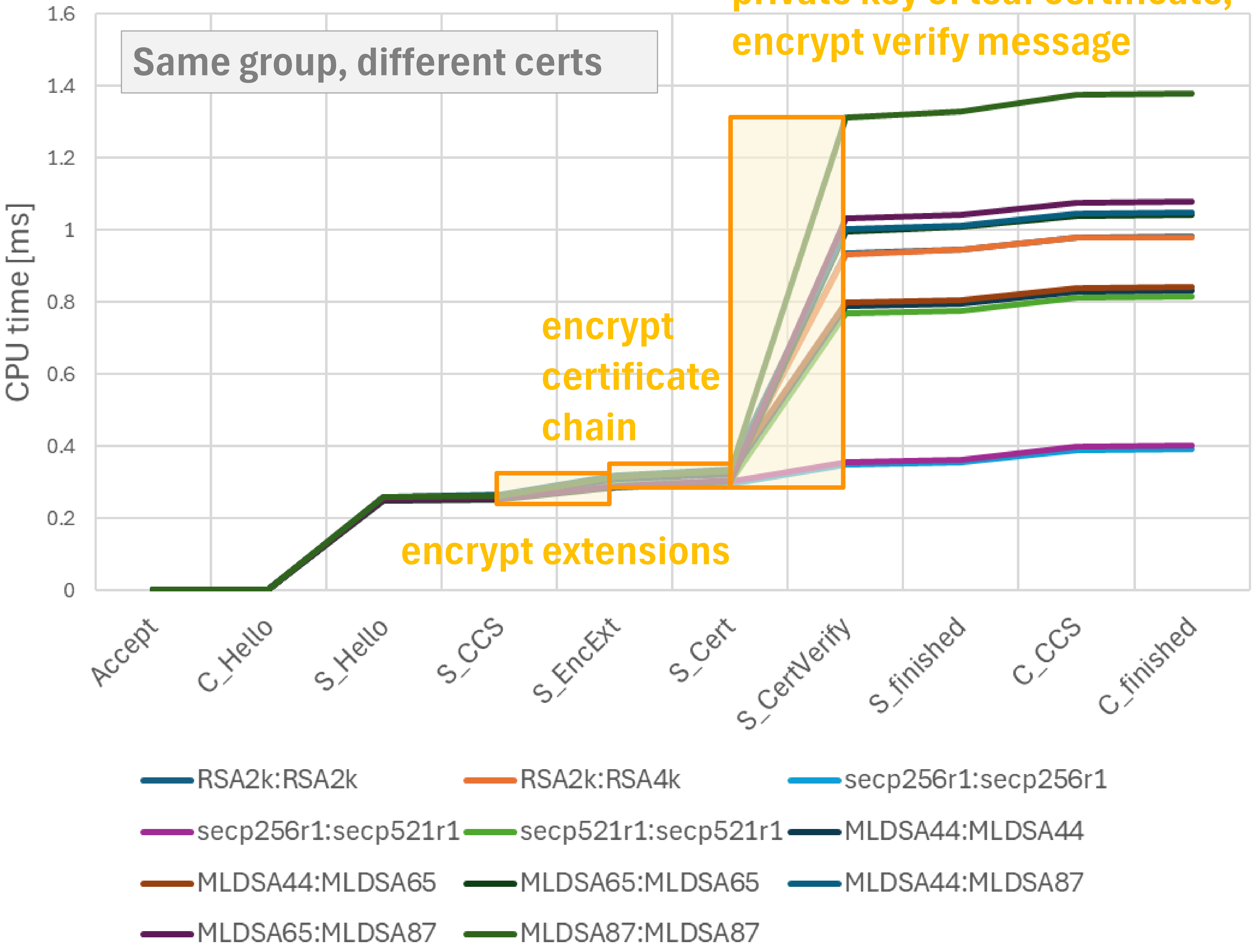
Note: For illustrative purposes only, generated with OpenSSL v3.5.0

Data extraction: SERVER

Constant:
2-stage cert chain MLDSA87:MLDSA87
TLS_AES_128_GCM_SHA256
Root certificate sent but not verified



Constant:
X25519MLKEM768
TLS_AES_128_GCM_SHA256
Root certificate sent but not verified

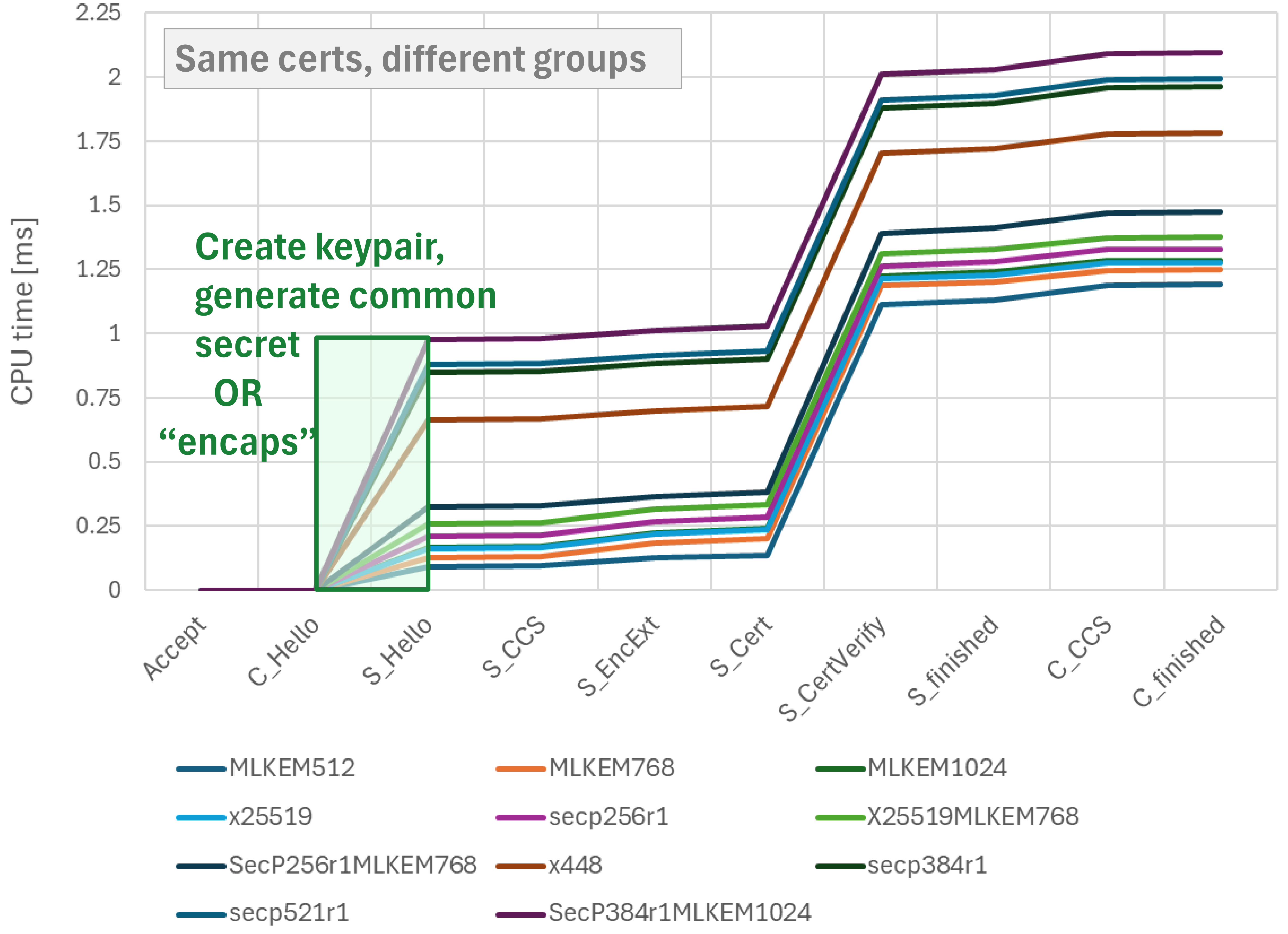


Certificate notation: <Leaf cert>:<Root cert>

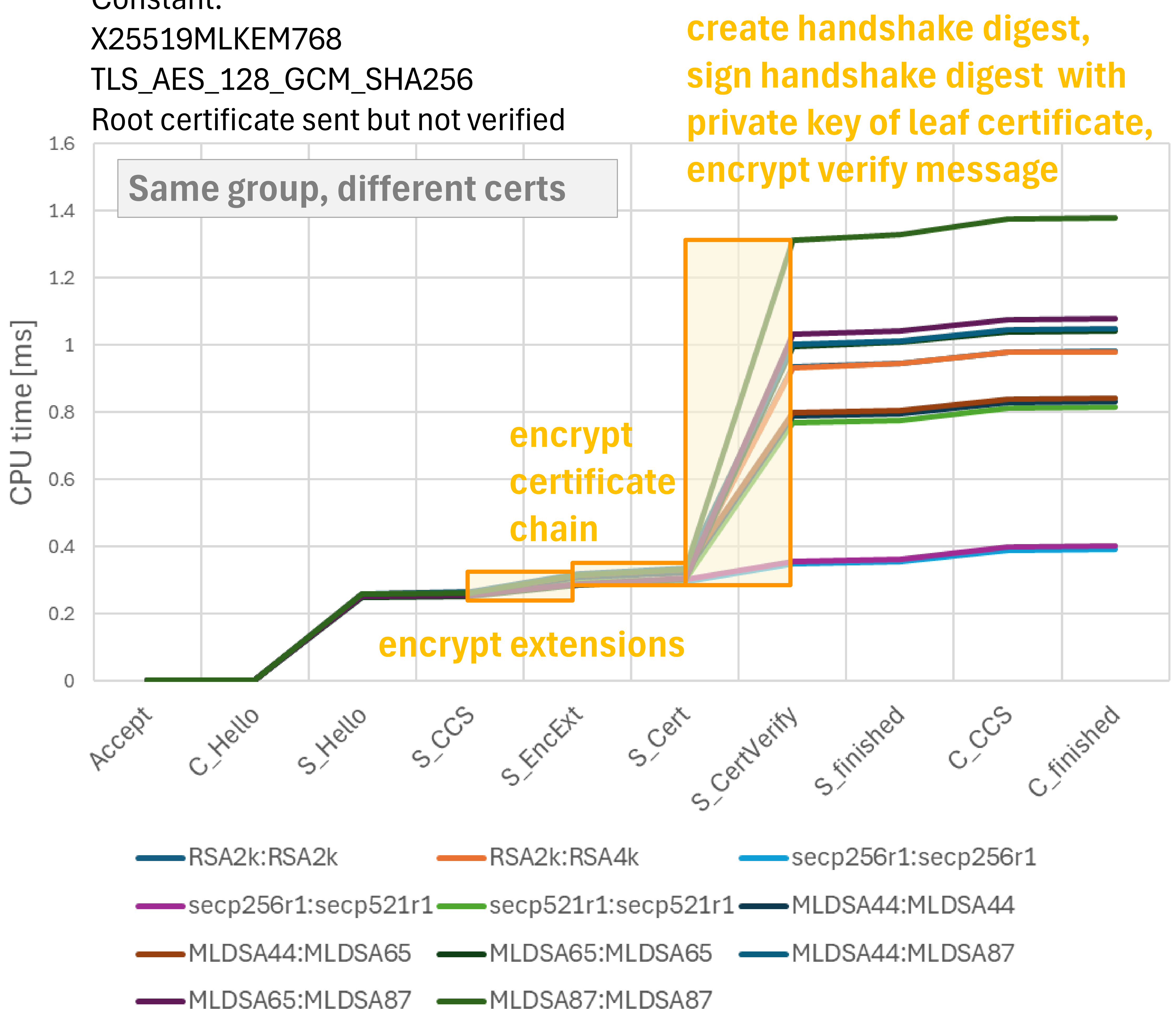
Note: For illustrative purposes only, generated with OpenSSL v3.5.0

Data extraction: SERVER

Constant:
2-stage cert chain MLDSA87:MLDSA87
TLS_AES_128_GCM_SHA256
Root certificate sent but not verified



Constant:
X25519MLKEM768
TLS_AES_128_GCM_SHA256
Root certificate sent but not verified



Certificate notation: <Leaf cert>:<Root cert>

Note: For illustrative purposes only, generated with OpenSSL v3.5.0

Measurement Setup

Hardware:

Bare metal rack server with Dual Intel(R) Xeon(R) “Ice Lake” Gold 6338 CPU @ 2.00GHz, 512 GB → Total 64C/128T
C-states & “Turbo” OFF

Setup:

TLS client & TLS server in c-code

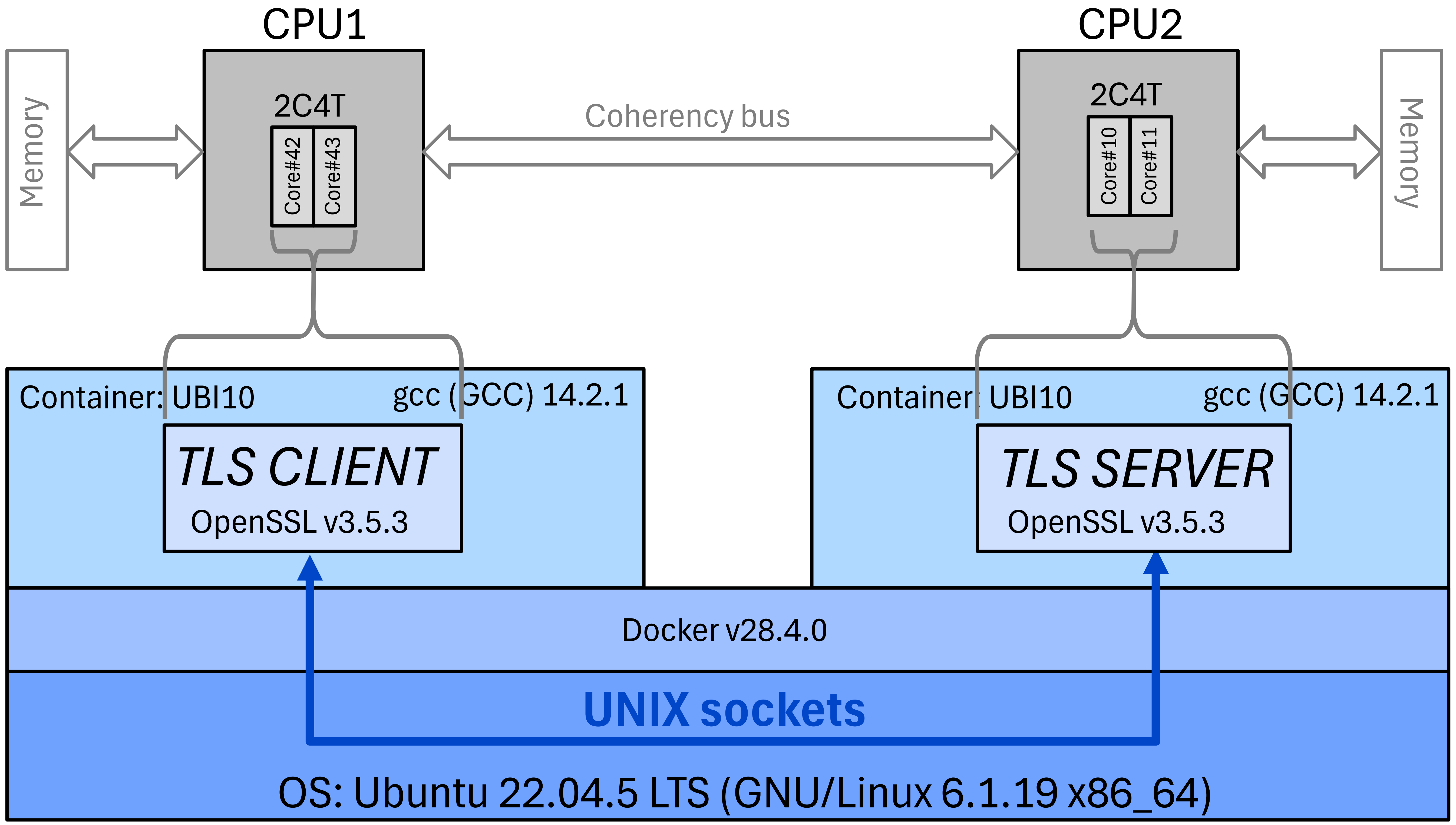
- on same bare metal computer
- each running on 2C4T (CPU pinning to two neighboring cores)
- each on its ‘own’ CPU chip
- each in its own Docker container
- sequential single TLS sessions
- communication via UNIX sockets
- single supported group and single keyshare in ClientHello → no HRR, no encExt

OpenSSL build options:

```
--prefix=/usr --with-rand-seed=rdcpu,os -DFAST_PCLMUL -DO3
-DECP_NISTZ256_ASM -DX25519_ASM -DCHACHA_ASM -POLY1305_ASM
-DOPENSSL_BN_ASM_MONT -DSHA1_ASM -DSHA256_ASM -DSHA512_ASM
-DKECCAK1600_ASM -DAES_ASM -DVPAES_ASM enable-tls1_3
enable-ec_nistp_64_gcc_128 no-ssl no-tls1 no-tls1_1
no-afalgeng no-tests shared threads -lm -mrdrnd
```

Experiment duration:

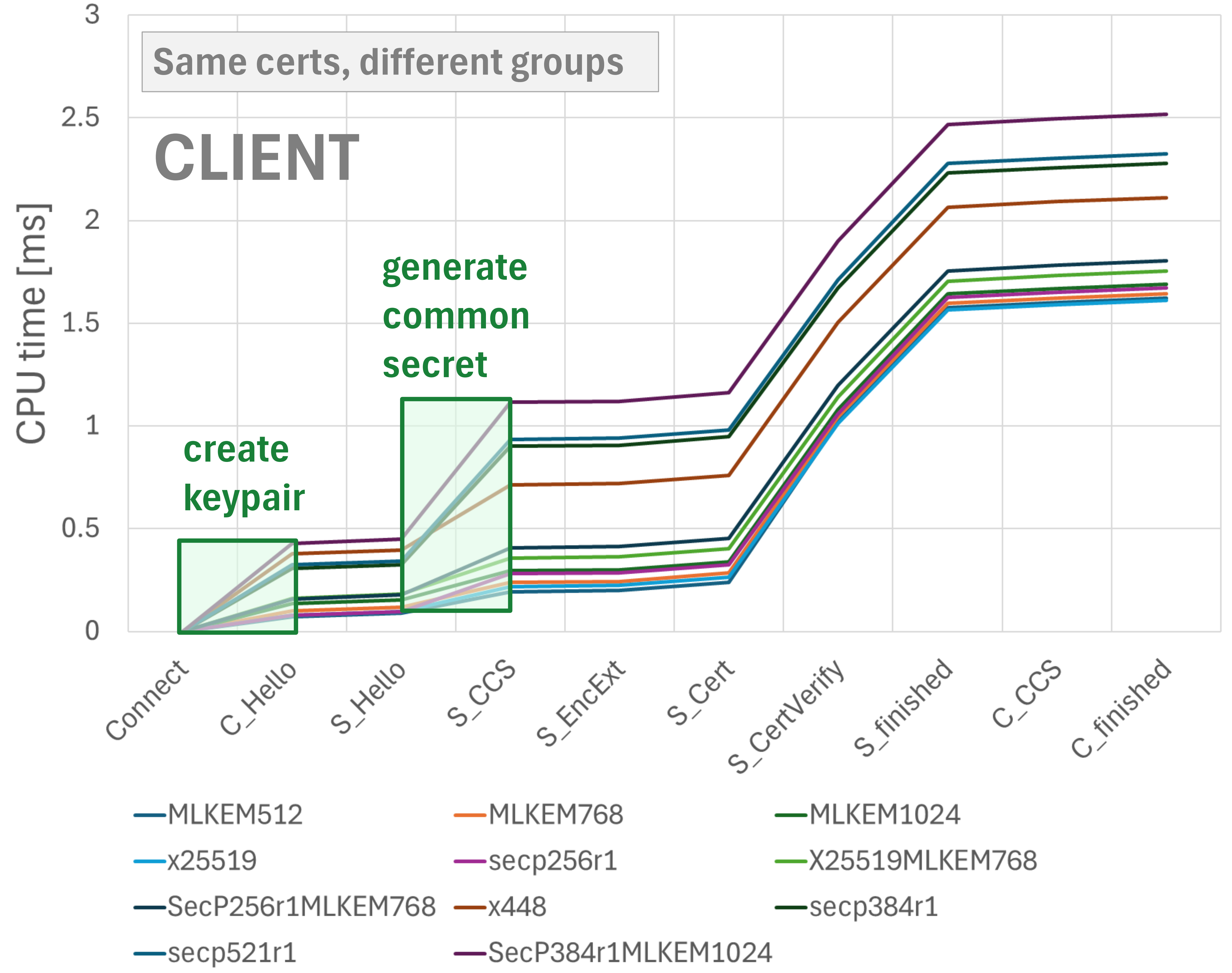
Each experiment with 10’000 warmup loops and 100’000 measurement loops
Measurement results show the 50% quantile (median) unless otherwise noted
Repeatability verified to be within ±1%



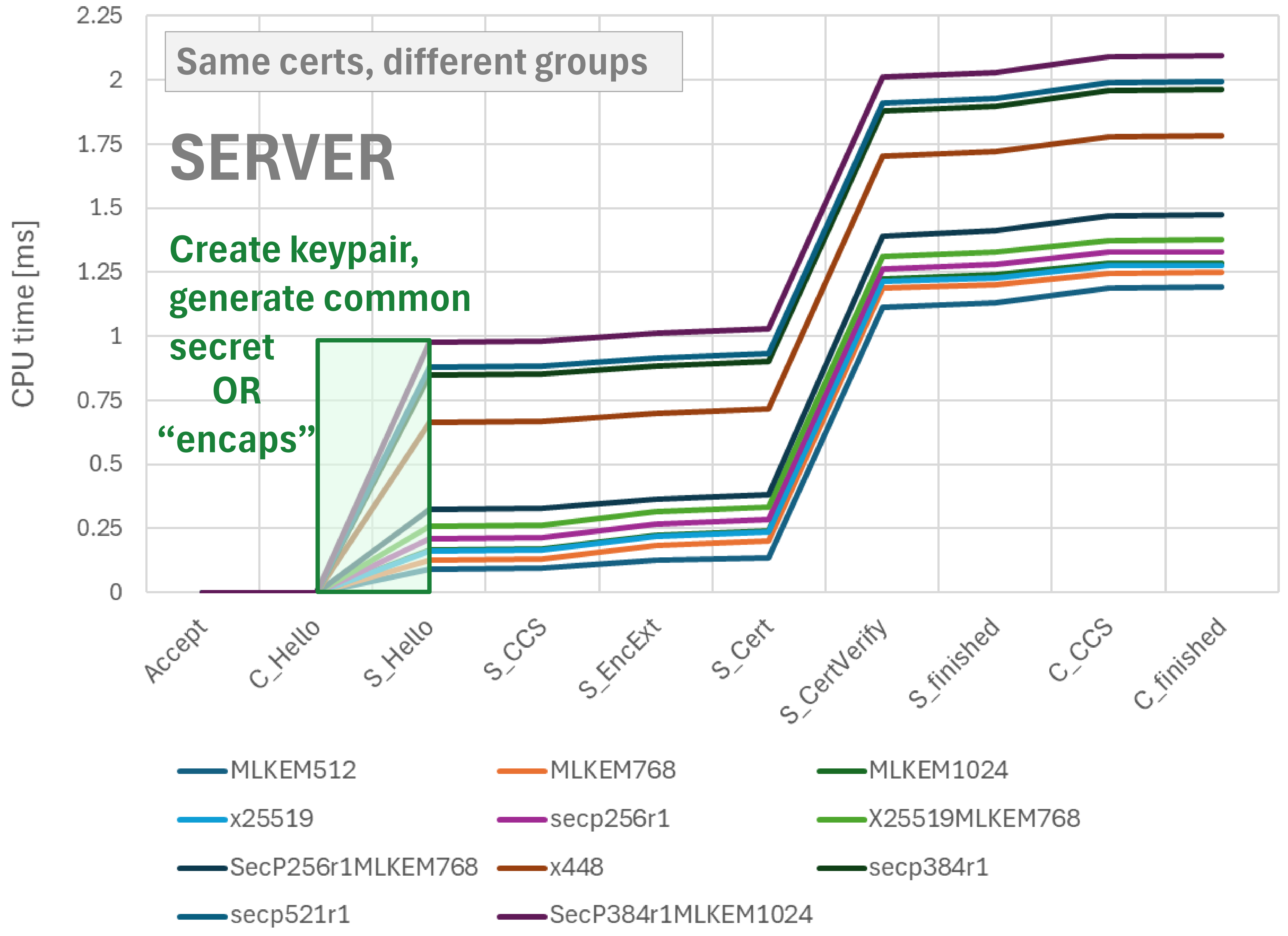
Remark: We did not see any impact on CPU times from using UNIX sockets or TCP via localhost

Measured Results: Key Agreement

Constant:
2-stage cert chain MLDSA87:MLDSA87
TLS_AES_128_GCM_SHA256
Root certificate sent but not verified

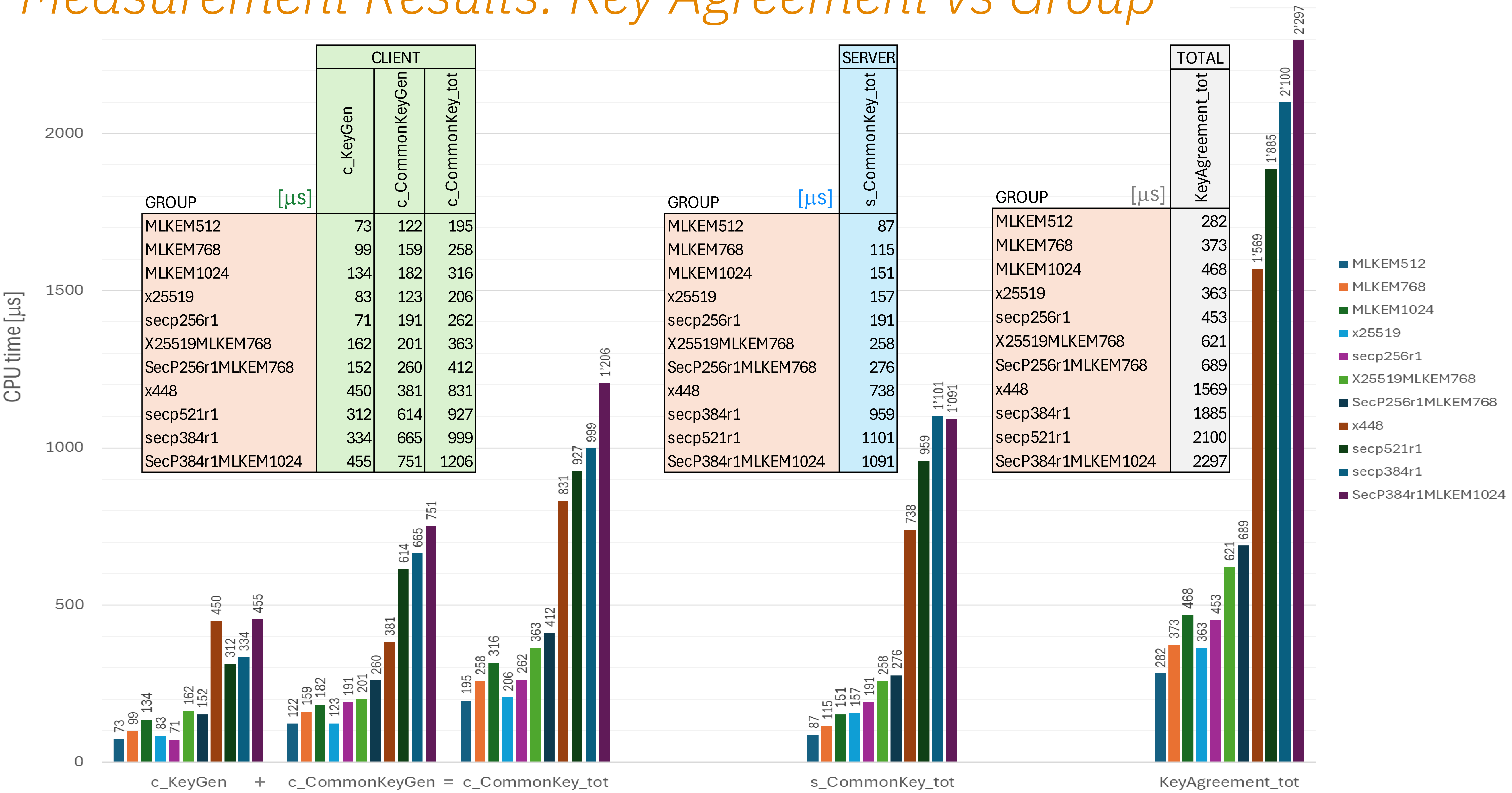


Constant:
2-stage cert chain MLDSA87:MLDSA87
TLS_AES_128_GCM_SHA256
Root certificate sent but not verified



Note: For illustrative purposes only, generated with OpenSSL v3.5.0

Measurement Results: Key Agreement vs Group



CLIENT

GROUP [µs]	c_KeyGen	c_CommonKeyGen	c_CommonKey_tot
MLKEM512	73	122	195
MLKEM768	99	159	258
MLKEM1024	134	182	316
x25519	83	123	206
secp256r1	71	191	262
X25519MLKEM768	162	201	363
SecP256r1MLKEM768	152	260	412
x448	450	381	831
secp521r1	312	614	927
secp384r1	334	665	999
SecP384r1MLKEM1024	455	751	1206

SERVER

GROUP [µs]	s_CommonKey_tot
MLKEM512	87
MLKEM768	115
MLKEM1024	151
x25519	157
secp256r1	191
X25519MLKEM768	258
SecP256r1MLKEM768	276
x448	738
secp384r1	959
secp521r1	1101
SecP384r1MLKEM1024	1091

TOTAL

GROUP [µs]	KeyAgreement_tot
MLKEM512	282
MLKEM768	373
MLKEM1024	468
x25519	363
secp256r1	453
X25519MLKEM768	621
SecP256r1MLKEM768	689
x448	1569
secp384r1	1885
secp521r1	2100
SecP384r1MLKEM1024	2297

CLIENT **SERVER** **TOTAL**

Measurement Results: Key Agreement vs Group

CLIENT

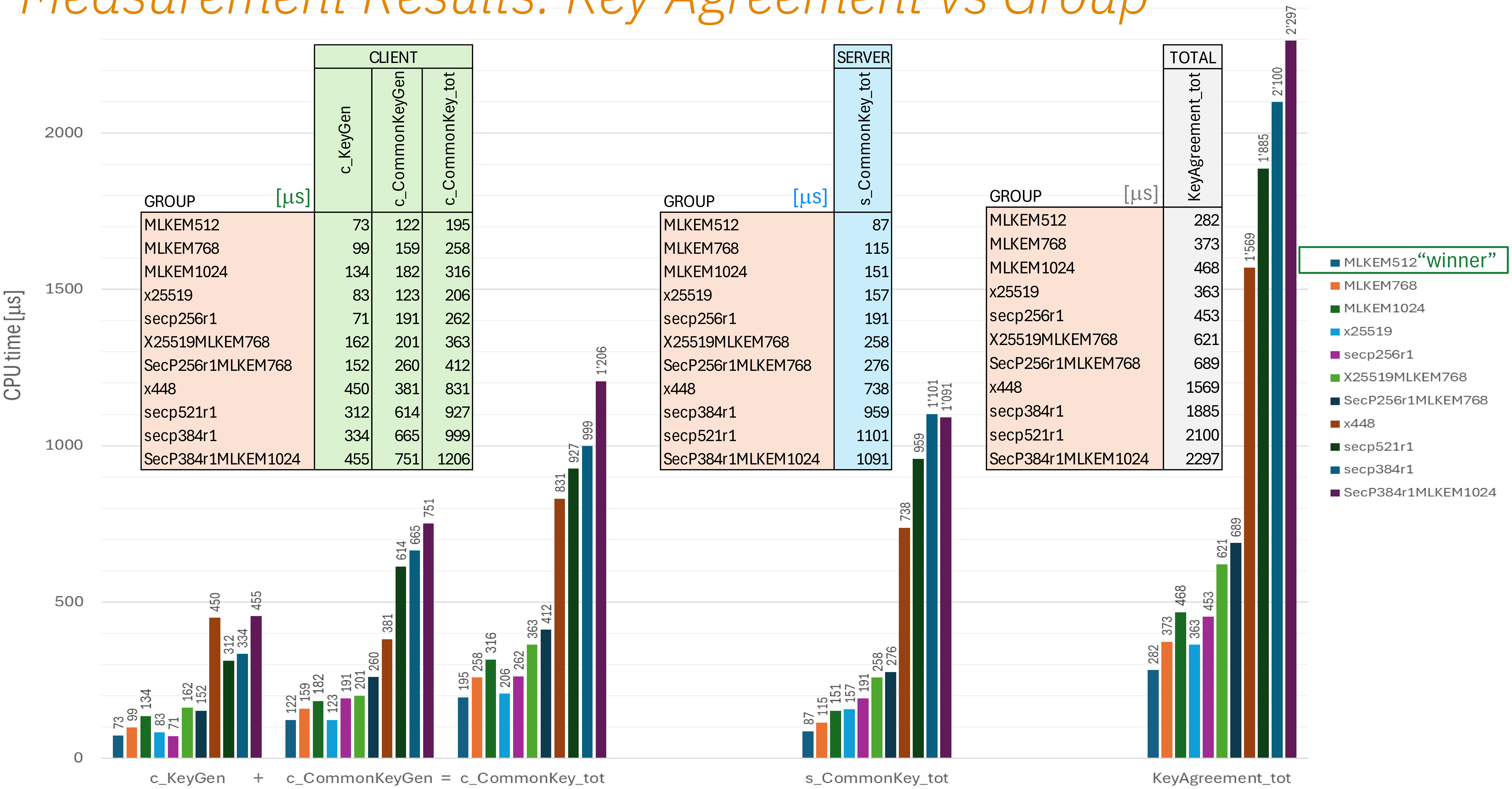
GROUP [µs]	c_KeyGen	c_CommonKeyGen	c_CommonKey_tot
MLKEM512	73	122	195
MLKEM768	99	159	258
MLKEM1024	134	182	316
x25519	83	123	206
secp256r1	71	191	262
X25519MLKEM768	162	201	363
SecP256r1MLKEM768	152	260	412
x448	450	381	831
secp521r1	312	614	927
secp384r1	334	665	999
SecP384r1MLKEM1024	455	751	1206

SERVER

GROUP [µs]	s_CommonKey_tot
MLKEM512	87
MLKEM768	115
MLKEM1024	151
x25519	157
secp256r1	191
X25519MLKEM768	258
SecP256r1MLKEM768	276
x448	738
secp384r1	959
secp521r1	1101
SecP384r1MLKEM1024	1091

TOTAL

GROUP [µs]	KeyAgreement_tot
MLKEM512	282
MLKEM768	373
MLKEM1024	468
x25519	363
secp256r1	453
X25519MLKEM768	621
SecP256r1MLKEM768	689
x448	1569
secp384r1	1885
secp521r1	2100
SecP384r1MLKEM1024	2297



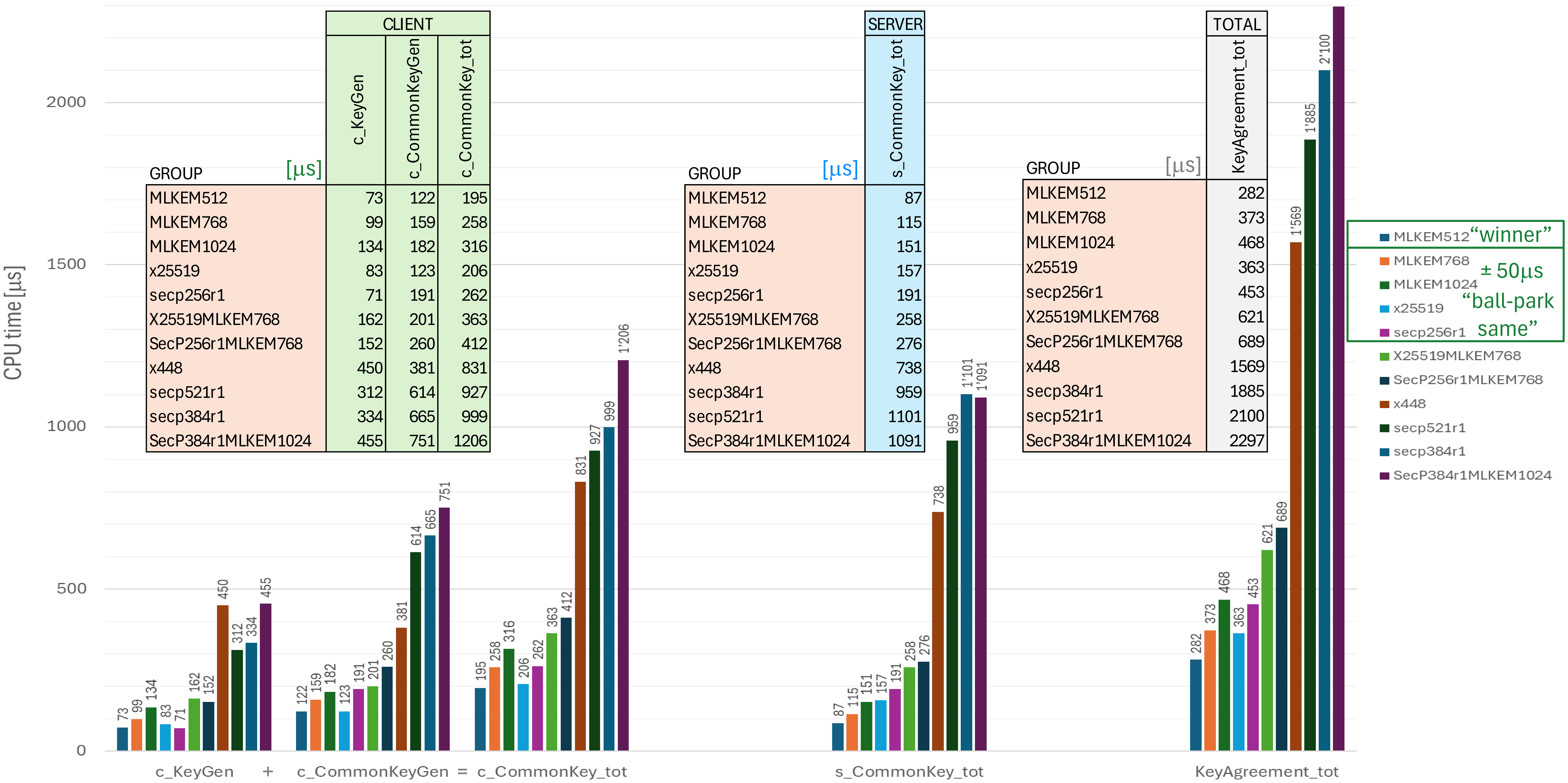
CLIENT **SERVER** **TOTAL**

Measurement Results: Key Agreement vs Group

GROUP [µs]	CLIENT c_KeyGen	CLIENT c_CommonKeyGen	CLIENT c_CommonKey_tot
MLKEM512	73	122	195
MLKEM768	99	159	258
MLKEM1024	134	182	316
x25519	83	123	206
secp256r1	71	191	262
X25519MLKEM768	162	201	363
SecP256r1MLKEM768	152	260	412
x448	450	381	831
secp521r1	312	614	927
secp384r1	334	665	999
SecP384r1MLKEM1024	455	751	1206

GROUP [µs]	SERVER s_CommonKey_tot
MLKEM512	87
MLKEM768	115
MLKEM1024	151
x25519	157
secp256r1	191
X25519MLKEM768	258
SecP256r1MLKEM768	276
x448	738
secp384r1	959
secp521r1	1101
SecP384r1MLKEM1024	1091

GROUP [µs]	TOTAL KeyAgreement_tot
MLKEM512	282
MLKEM768	373
MLKEM1024	468
x25519	363
secp256r1	453
X25519MLKEM768	621
SecP256r1MLKEM768	689
x448	1569
secp384r1	1885
secp521r1	2100
SecP384r1MLKEM1024	2297



CLIENT **SERVER** **TOTAL**

Measurement Results: Key Agreement vs Group

CLIENT

GROUP [µs]	c_KeyGen	c_CommonKeyGen	c_CommonKey_tot
MLKEM512	73	122	195
MLKEM768	99	159	258
MLKEM1024	134	182	316
x25519	83	123	206
secp256r1	71	191	262
X25519MLKEM768	162	201	363
SecP256r1MLKEM768	152	260	412
x448	450	381	831
secp521r1	312	614	927
secp384r1	334	665	999
SecP384r1MLKEM1024	455	751	1206

SERVER

GROUP [µs]	s_CommonKey_tot
MLKEM512	87
MLKEM768	115
MLKEM1024	151
x25519	157
secp256r1	191
X25519MLKEM768	258
SecP256r1MLKEM768	276
x448	738
secp384r1	959
secp521r1	1101
SecP384r1MLKEM1024	1091

TOTAL

GROUP [µs]	KeyAgreement_tot
MLKEM512	282
MLKEM768	373
MLKEM1024	468
x25519	363
secp256r1	453
X25519MLKEM768	621
SecP256r1MLKEM768	689
x448	1569
secp384r1	1885
secp521r1	2100
SecP384r1MLKEM1024	2297

CLIENT **SERVER** **TOTAL**

Measurement Results: Key Agreement vs Group

GROUP [µs]	CLIENT c_KeyGen	CLIENT c_CommonKeyGen	CLIENT c_CommonKey_tot
MLKEM512	73	122	195
MLKEM768	99	159	258
MLKEM1024	134	182	316
x25519	83	123	206
secp256r1	71	191	262
X25519MLKEM768	162	201	363
SecP256r1MLKEM768	152	260	412
x448	450	381	831
secp521r1	312	614	927
secp384r1	334	665	999
SecP384r1MLKEM1024	455	751	1206

GROUP [µs]	SERVER s_CommonKey_tot
MLKEM512	87
MLKEM768	115
MLKEM1024	151
x25519	157
secp256r1	191
X25519MLKEM768	258
SecP256r1MLKEM768	276
x448	738
secp384r1	959
secp521r1	1101
SecP384r1MLKEM1024	1091

GROUP [µs]	TOTAL KeyAgreement_tot
MLKEM512	282
MLKEM768	373
MLKEM1024	468
x25519	363
secp256r1	453
X25519MLKEM768	621
SecP256r1MLKEM768	689
x448	1569
secp384r1	1885
secp521r1	2100
SecP384r1MLKEM1024	2297

Measurement Results: Key Agreement vs Group

GROUP [µs]	CLIENT c_KeyGen	CLIENT c_CommonKeyGen	CLIENT c_CommonKey_tot
MLKEM512	73	122	195
MLKEM768	99	159	258
MLKEM1024	134	182	316
x25519	83	123	206
secp256r1	71	191	262
X25519MLKEM768	162	201	363
SecP256r1MLKEM768	152	260	412
x448	450	381	831
secp521r1	312	614	927
secp384r1	334	665	999
SecP384r1MLKEM1024	455	751	1206

GROUP [µs]	SERVER s_CommonKey_tot
MLKEM512	87
MLKEM768	115
MLKEM1024	151
x25519	157
secp256r1	191
X25519MLKEM768	258
SecP256r1MLKEM768	276
x448	738
secp384r1	959
secp521r1	1101
SecP384r1MLKEM1024	1091

GROUP [µs]	TOTAL KeyAgreement_tot
MLKEM512	282
MLKEM768	373
MLKEM1024	468
x25519	363
secp256r1	453
X25519MLKEM768	621
SecP256r1MLKEM768	689
x448	1569
secp384r1	1885
secp521r1	2100
SecP384r1MLKEM1024	2297

Measurement Results: Key Agreement vs Group

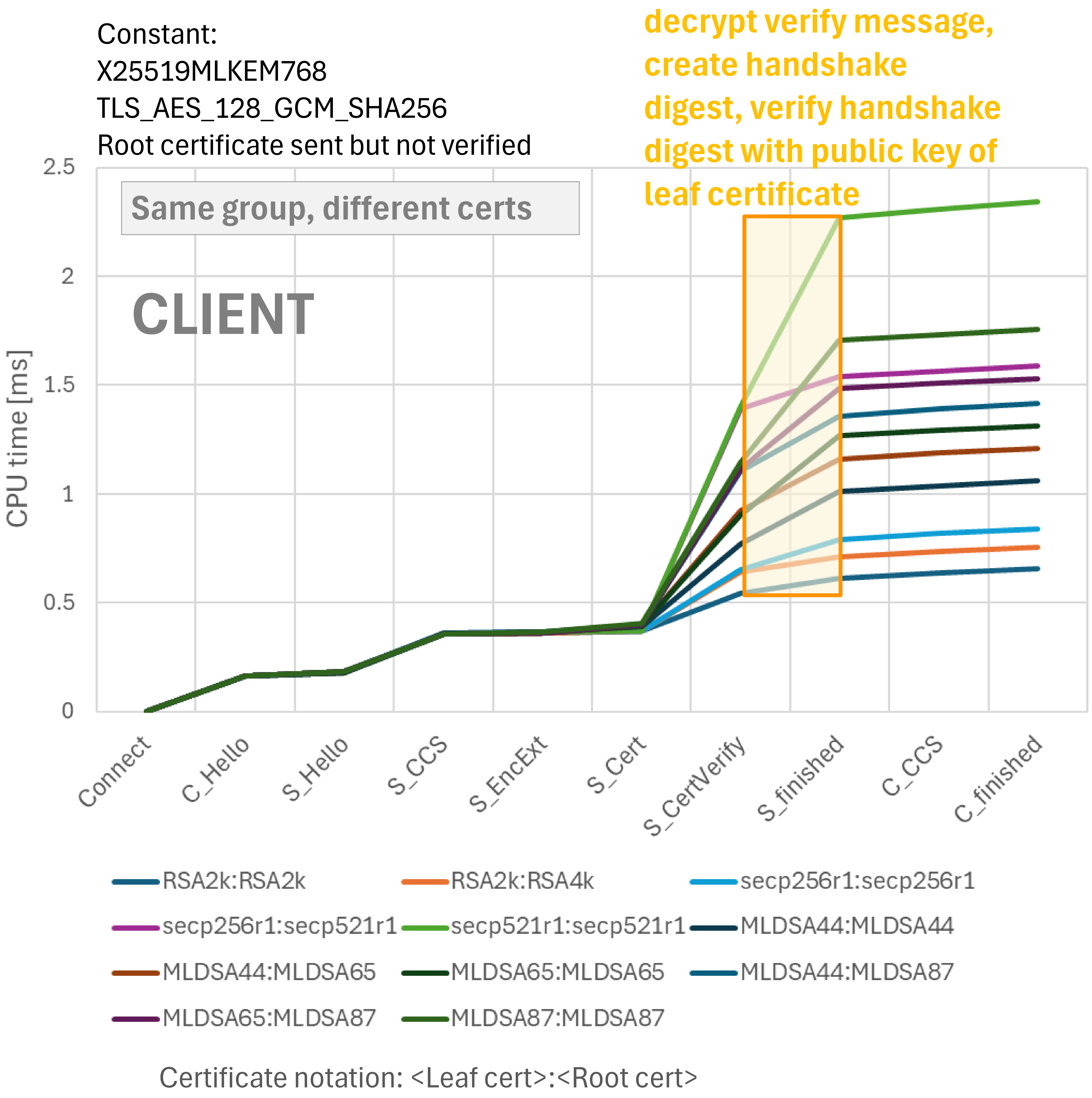
GROUP [µs]	CLIENT c_KeyGen	CLIENT c_CommonKeyGen	CLIENT c_CommonKey_tot
MLKEM512	73	122	195
MLKEM768	99	159	258
MLKEM1024	134	182	316
x25519	83	123	206
secp256r1	71	191	262
X25519MLKEM768	162	201	363
SecP256r1MLKEM768	152	260	412
x448	450	381	831
secp521r1	312	614	927
secp384r1	334	665	999
SecP384r1MLKEM1024	455	751	1206

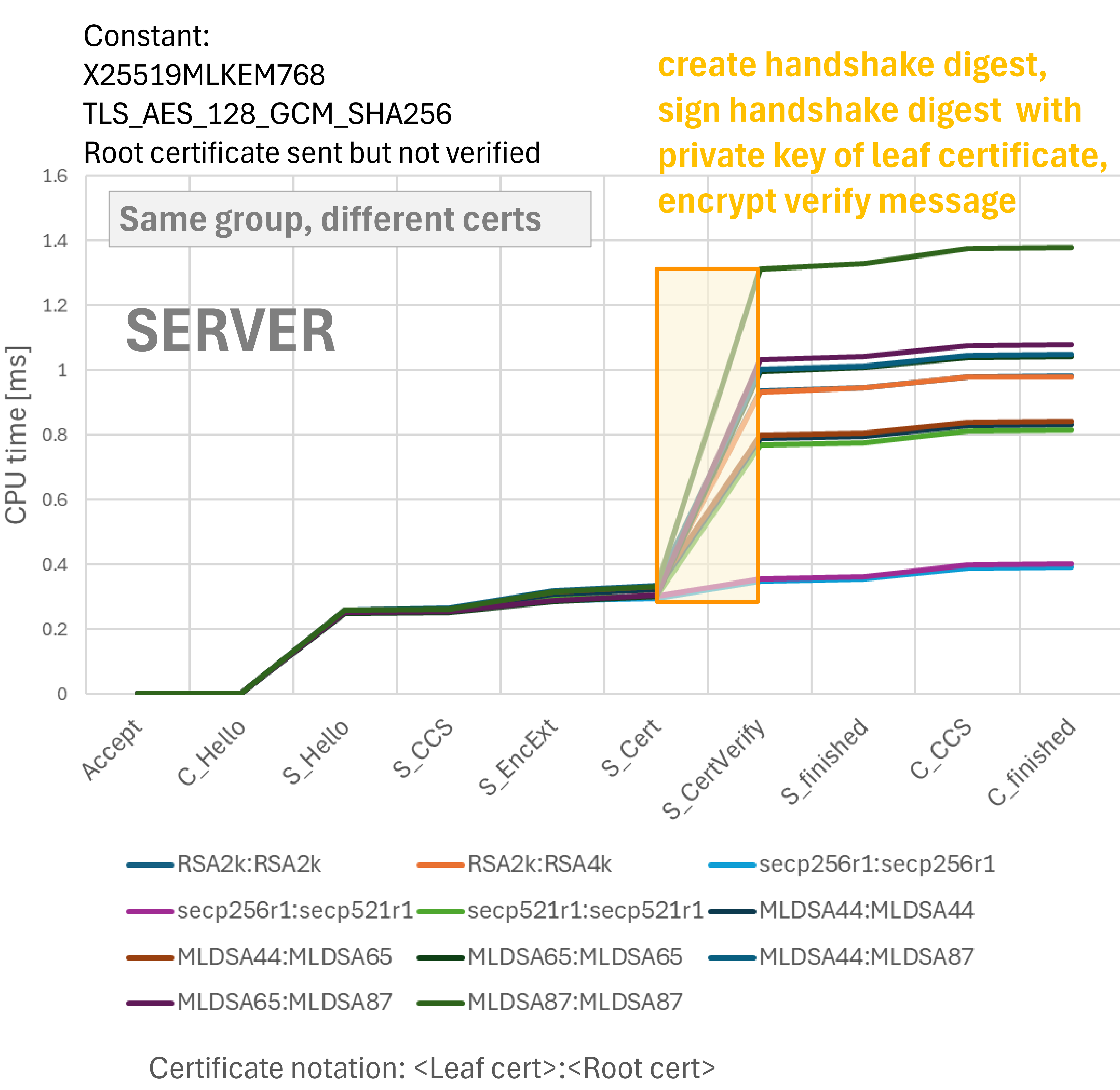
GROUP [µs]	SERVER s_CommonKey_tot
MLKEM512	87
MLKEM768	115
MLKEM1024	151
x25519	157
secp256r1	191
X25519MLKEM768	258
SecP256r1MLKEM768	276
x448	738
secp384r1	959
secp521r1	1101
SecP384r1MLKEM1024	1091

GROUP [µs]	TOTAL KeyAgreement_tot
MLKEM512	282
MLKEM768	373
MLKEM1024	468
x25519	363
secp256r1	453
X25519MLKEM768	621
SecP256r1MLKEM768	689
x448	1569
secp384r1	1885
secp521r1	2100
SecP384r1MLKEM1024	2297

CLIENT **SERVER** **TOTAL**

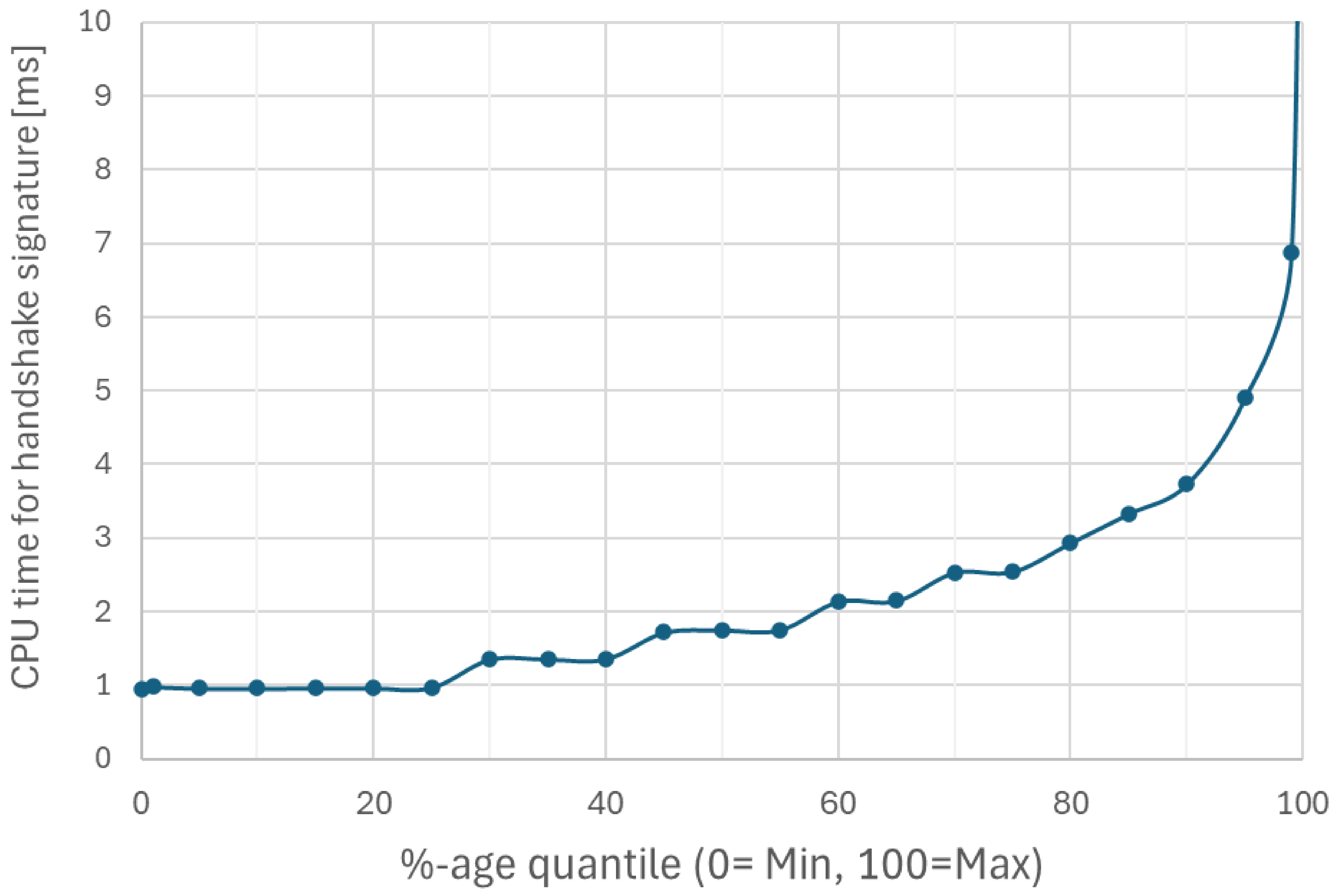
Measured Results: Sign/Verify Handshake Messages





Measurement Results: MLDSA signature subtlety

Group X25519
2-stage cert chain MLDSA87:MLDSA87
TLS_AES_128_GCM_SHA256
Root certificate sent but not verified

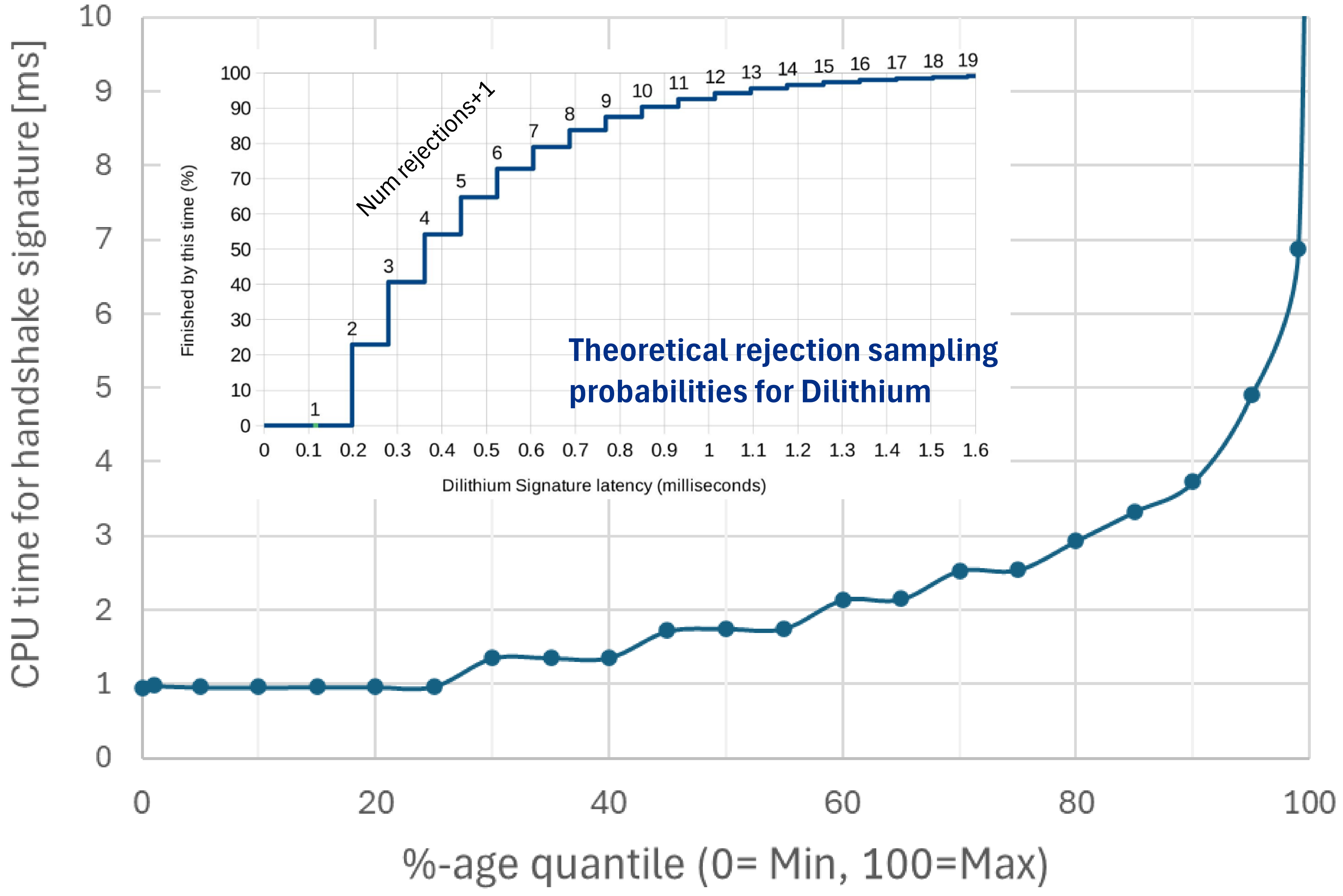


MLDSA** rejection sampling leads to **unbound latency variations and hence to ***tail latencies***

Rejection sampling is a security mechanism that ensures signatures are statistically close to secret-independent and corrects for potential leakage of secret information during signature generation.
When a generated signature does not meet certain criteria, it is rejected, and the process is repeated.

Measurement Results: MLDSA signature subtlety

Group X25519
2-stage cert chain MLDSA87:MLDSA87
TLS_AES_128_GCM_SHA256
Root certificate sent but not verified

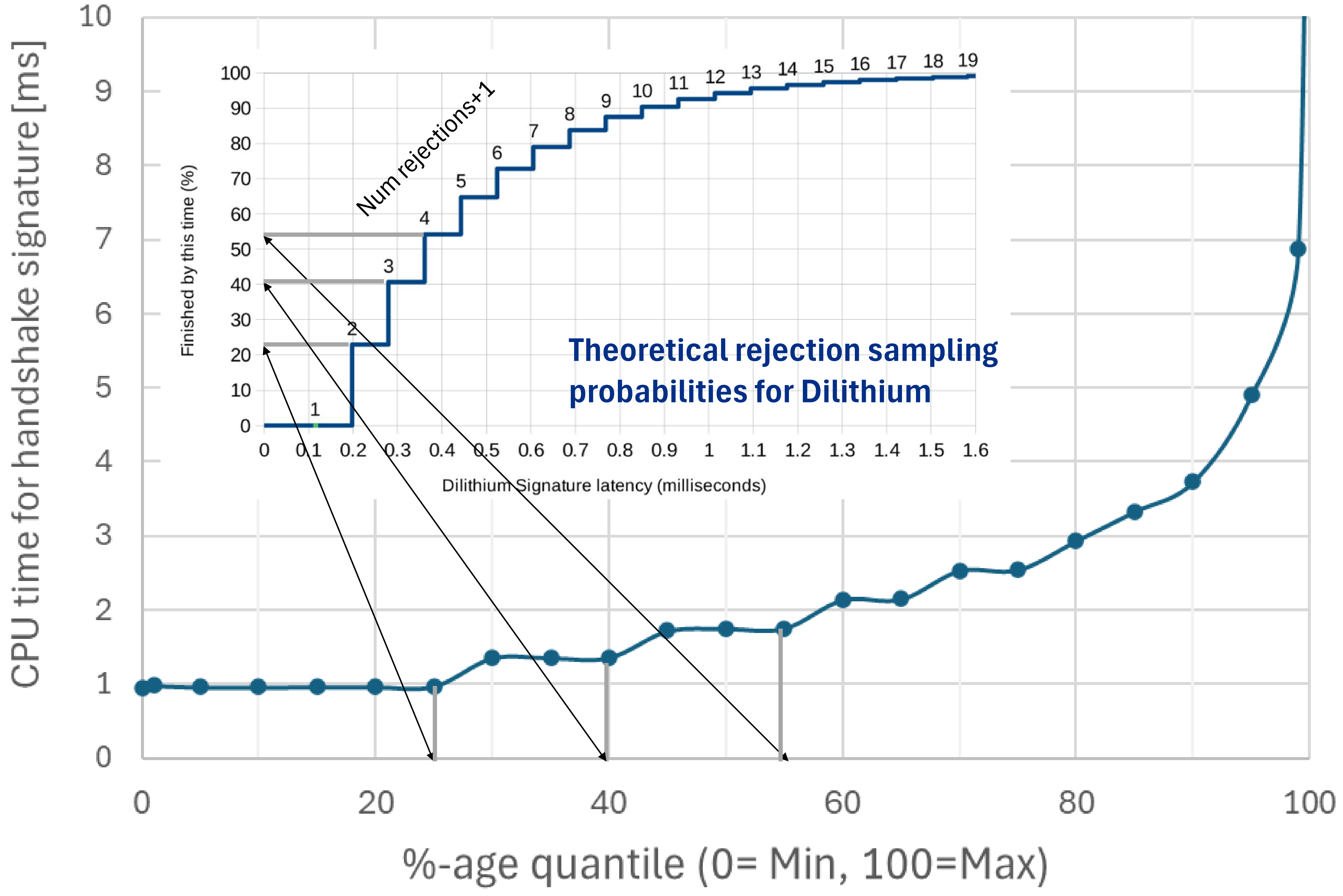


MLDSA** rejection sampling leads to **unbound latency variations and hence to ***tail latencies***

Rejection sampling is a security mechanism that ensures signatures are statistically close to secret-independent and corrects for potential leakage of secret information during signature generation.
When a generated signature does not meet certain criteria, it is rejected, and the process is repeated.

Measurement Results: MLDSA signature subtlety

Group X25519
2-stage cert chain MLDSA87:MLDSA87
TLS_AES_128_GCM_SHA256
Root certificate sent but not verified

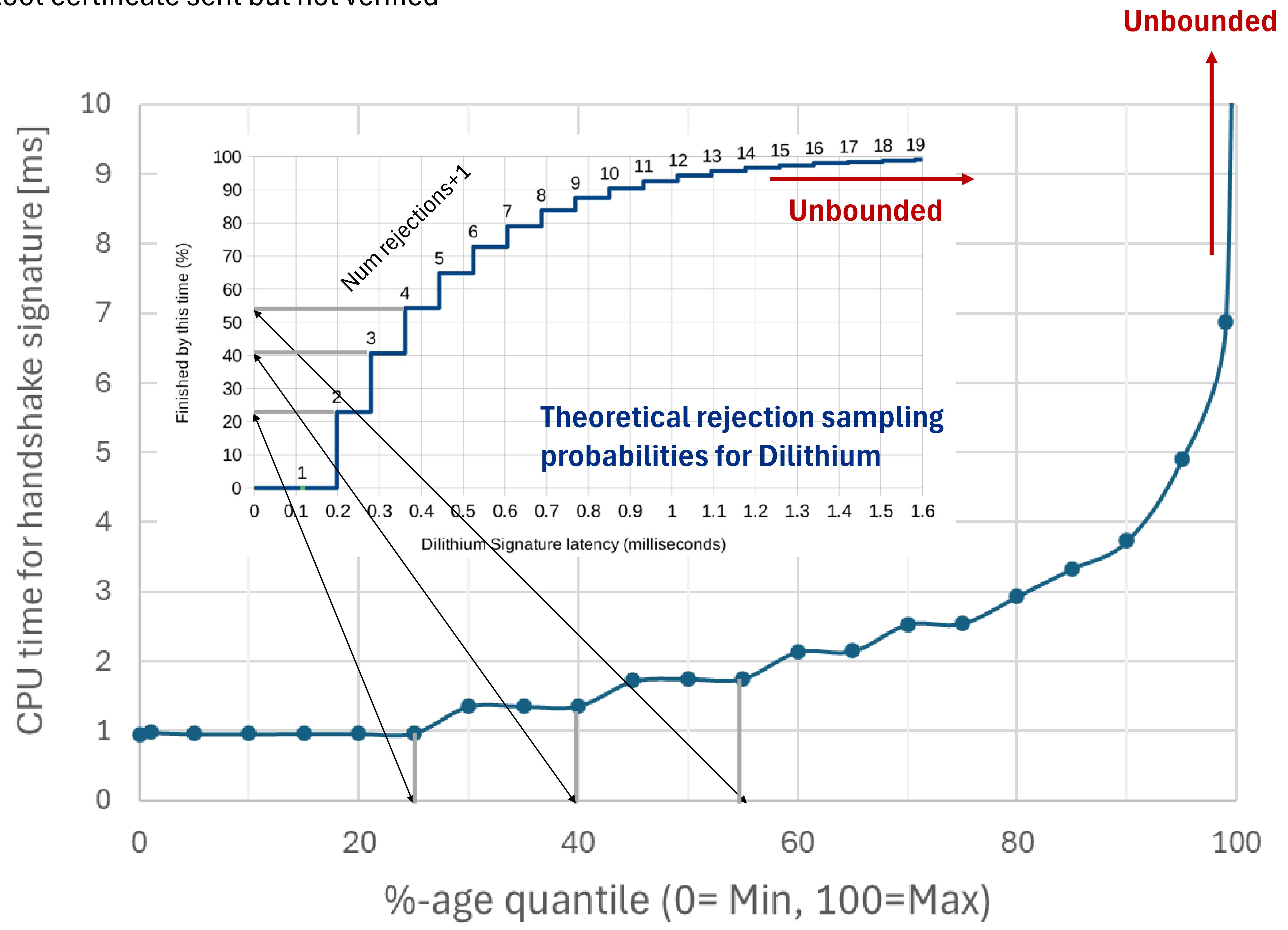


<u>MLDSA</u> rejection sampling leads to <u>unbound latency variations</u> and hence to ***<u>tail latencies</u>***

Rejection sampling is a security mechanism that ensures signatures are statistically close to secret-independent and corrects for potential leakage of secret information during signature generation.
When a generated signature does not meet certain criteria, it is rejected, and <u>the process is repeated</u>.

Measurement Results: MLDSA signature subtlety

Group X25519
2-stage cert chain MLDSA87:MLDSA87
TLS_AES_128_GCM_SHA256
Root certificate sent but not verified

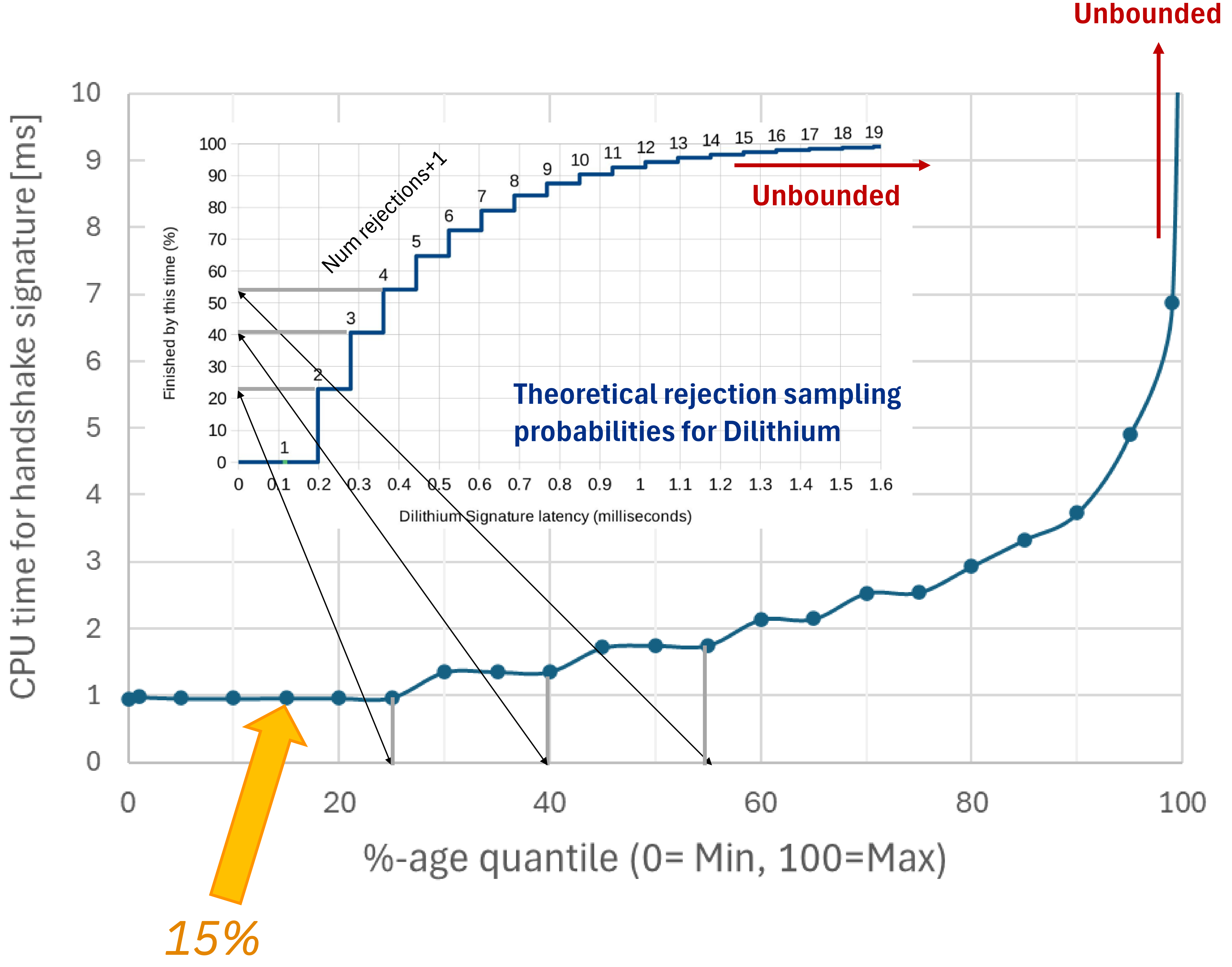


<u>MLDSA</u> rejection sampling leads to <u>unbound latency variations</u> and hence to ***<u>tail latencies</u>***

Rejection sampling is a security mechanism that ensures signatures are statistically close to secret-independent and corrects for potential leakage of secret information during signature generation.
When a generated signature does not meet certain criteria, it is rejected, and <u>the process is repeated</u>.

Measurement Results: MLDSA signature subtlety

Group X25519
2-stage cert chain MLDSA87:MLDSA87
TLS_AES_128_GCM_SHA256
Root certificate sent but not verified



<u>MLDSA</u> rejection sampling leads to <u>unbound latency variations</u> and hence to ***<u>tail latencies</u>***

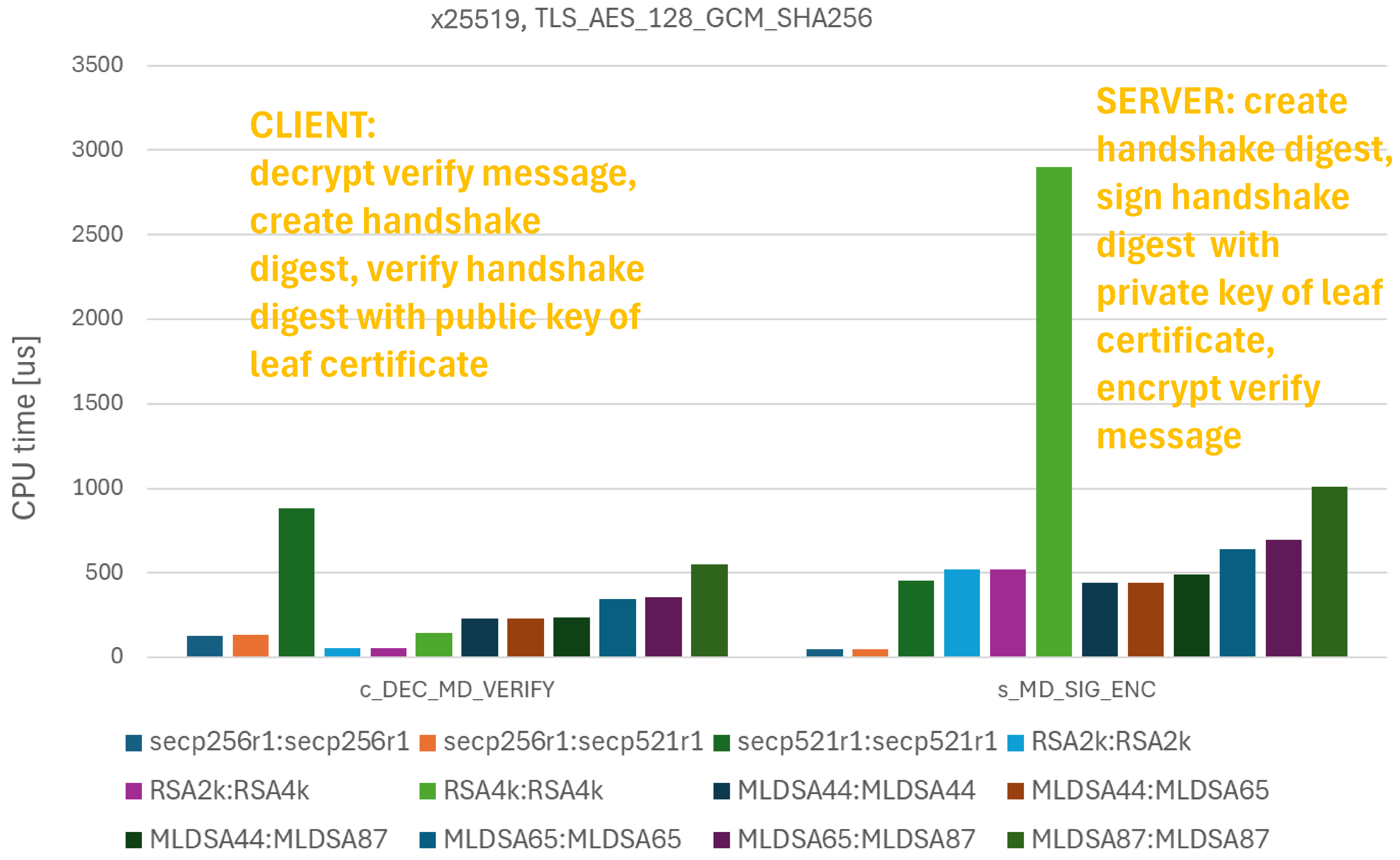
Rejection sampling is a security mechanism that ensures signatures are statistically close to secret-independent and corrects for potential leakage of secret information during signature generation.
When a generated signature does not meet certain criteria, it is rejected, and <u>the process is repeated</u>.

➔ *Can we have sign/verify algorithms without rejection sampling, please!*

➔ *We show results for the 15% quantile*

Measured Results: Sign/Verify Handshake Messages

HANDSHAKE SIGN/VERIFY CPU TIMES vs CERTIFICATE CHAINS

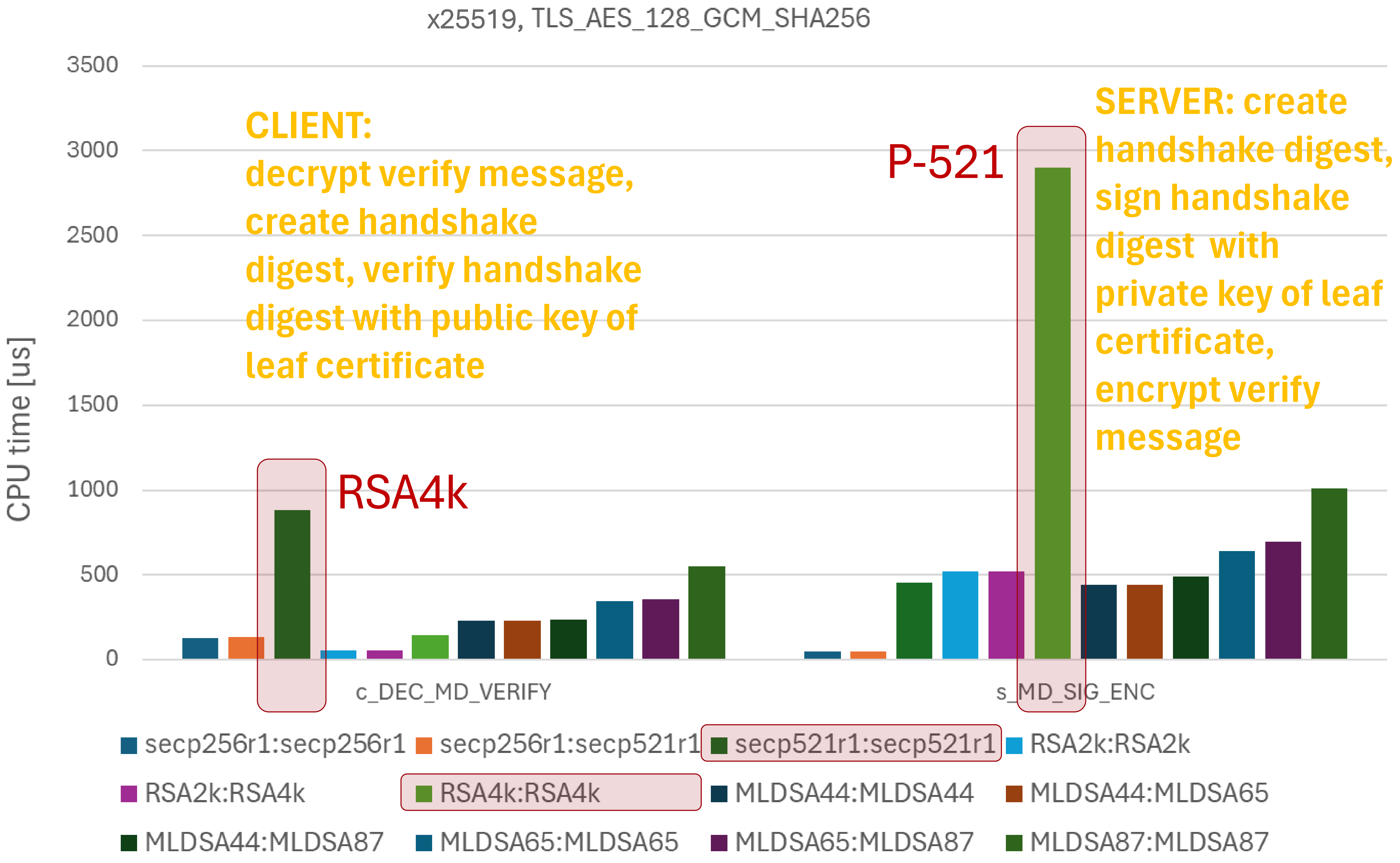


Only the crypto algorithm from the <u>leaf certificate</u> is used to sign/verify

Ciphers have a small influence (± 50 μs), hence we show results only for TLS_AES_128_GCM_SHA256

Measured Results: Sign/Verify Handshake Messages

HANDSHAKE SIGN/VERIFY CPU TIMES vs CERTIFICATE CHAINS

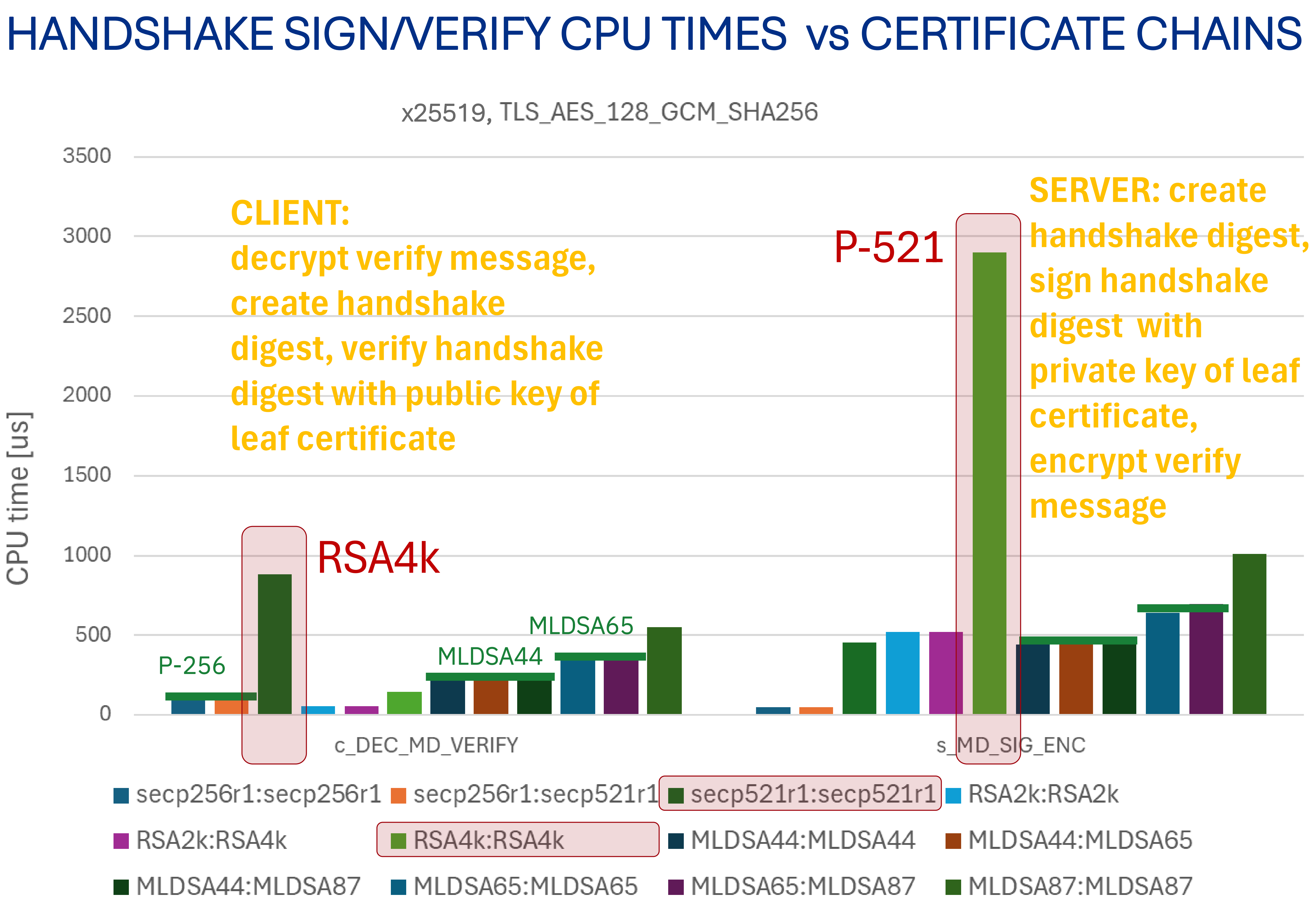


Only the crypto algorithm from the <u>leaf certificate</u> is used to sign/verify

Ciphers have a small influence (± 50 µs), hence we show results only for TLS_AES_128_GCM_SHA256

<u>Leaf certificates with RSA4k or Secp521r1 are very costly (up to +2 ms latency)</u>

Measured Results: Sign/Verify Handshake Messages



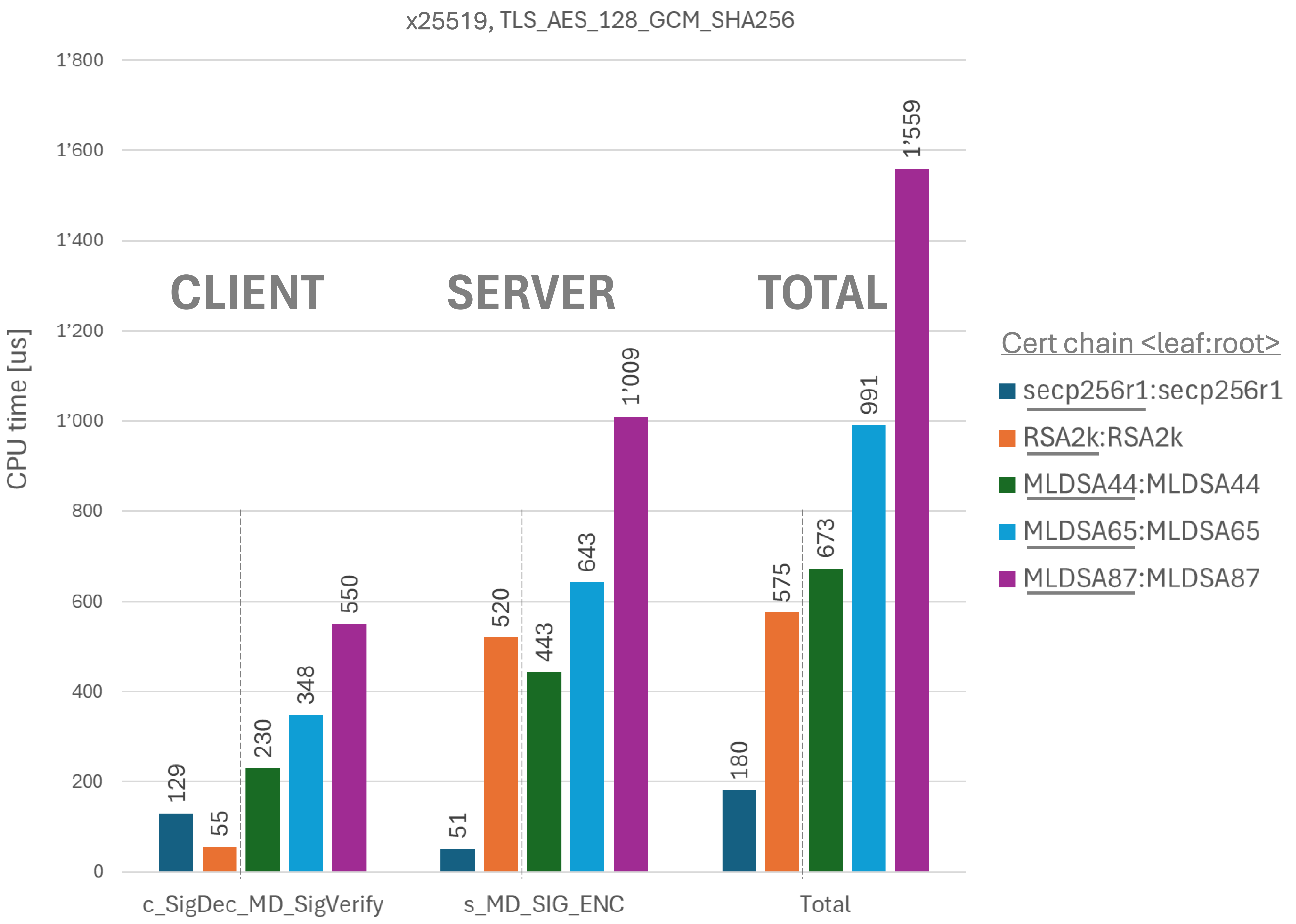
Only the crypto algorithm from the <u>leaf certificate</u> is used to sign/verify

Ciphers have a small influence (± 50 µs), hence we show results only for TLS_AES_128_GCM_SHA256

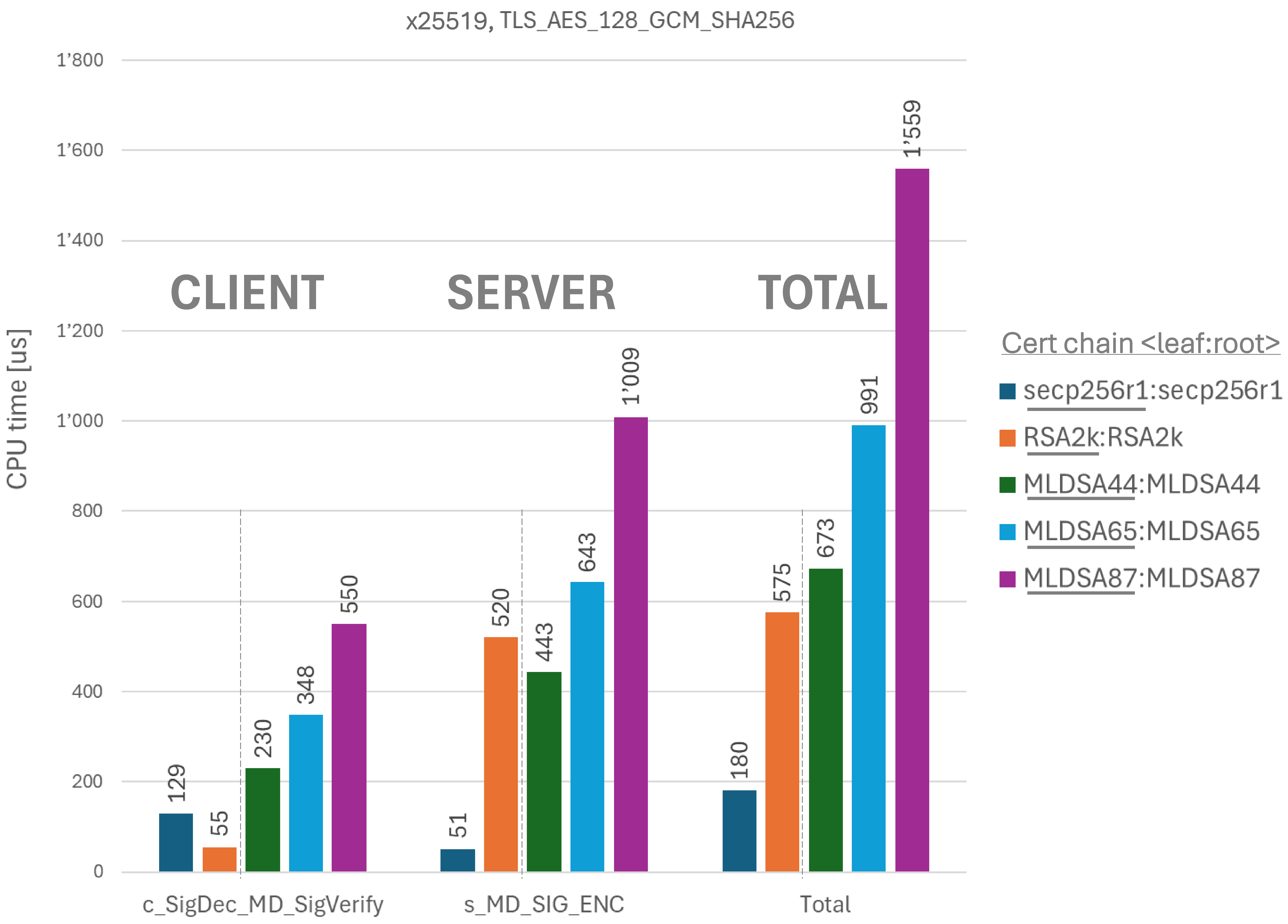
<u>Leaf certificates with RSA4k or Secp521r1 are very costly (up to +2 ms latency)</u>

The root certificate algorithm has no influence on the timings; hence we zoom in on the next chart.....

Measured Results: Sign/Verify Handshake Messages



Measured Results: Sign/Verify Handshake Messages



Client:
- RSA2k is best
- MLDSA[44|65|87] adds ~[200|300|400]µs

Server:
- Secp256r1 is best (by a large margin)
- MLDSA[44|65|87] adds ~[400|600|1’000]µs

Total:
- Secp256r1 is best (by a large margin)
- MLDSA[44|65|87] adds ~[500|800|1’400]µs

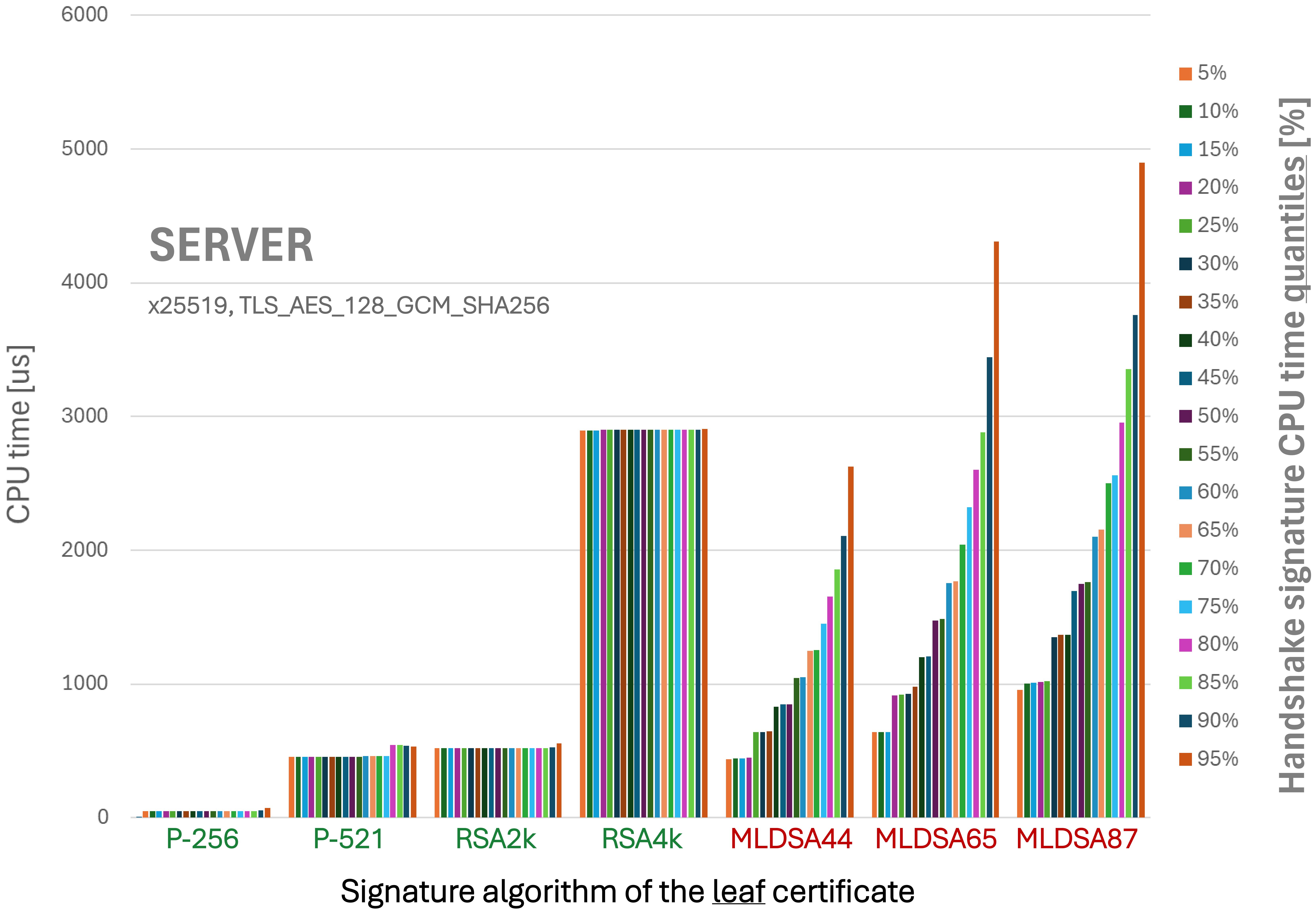
Measured Results: Sign/Verify Handshake Messages

Client:
- RSA2k is best
- MLDSA[44|65|87] adds ~[200|300|400]µs

Server:
- Secp256r1 is best (by a large margin)
- MLDSA[44|65|87] adds ~[400|600|1’000]µs

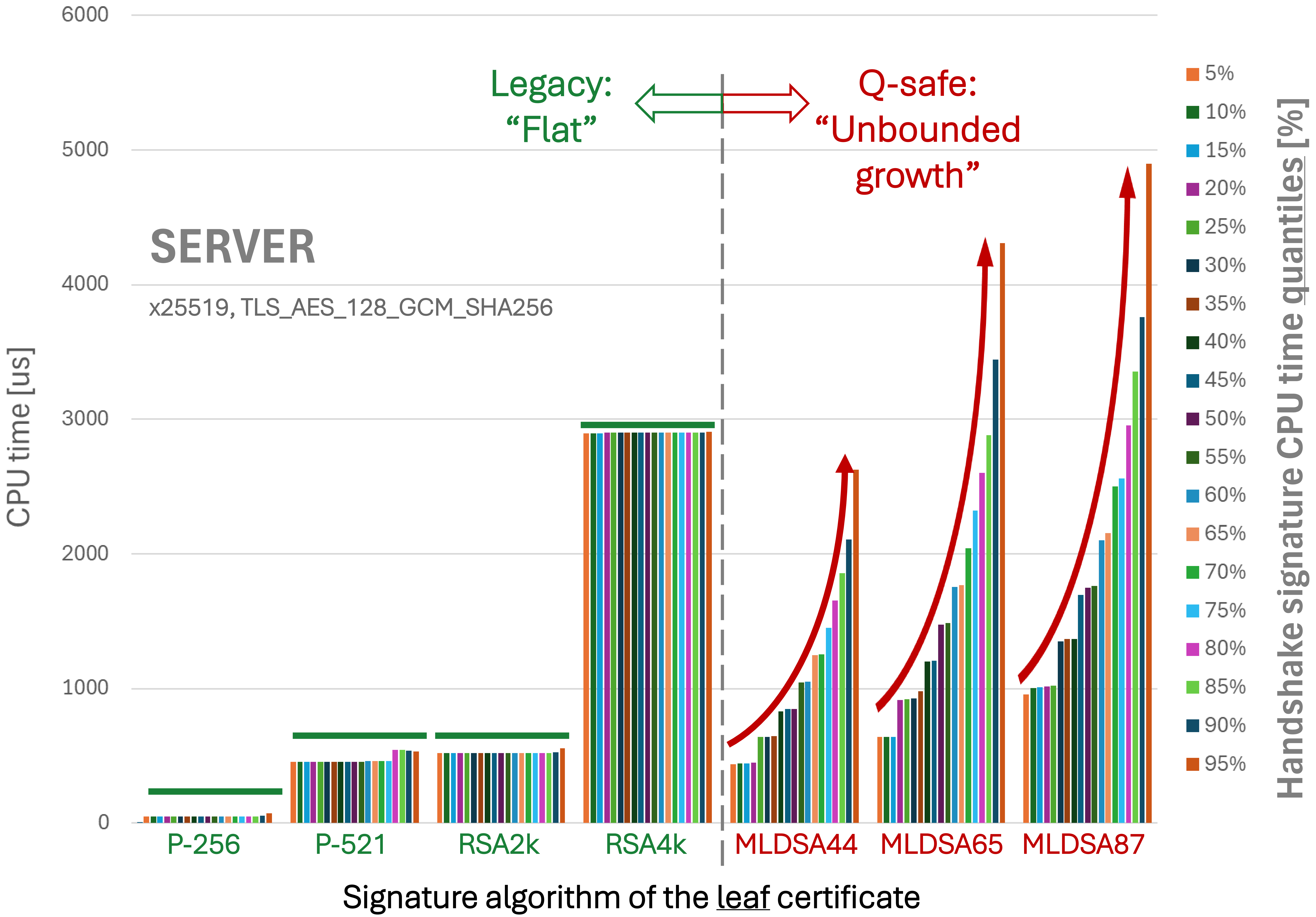
Total:
- Secp256r1 is best (by a large margin)
- MLDSA[44|65|87] adds ~[500|800|1’400]µs

Measured Results: <u>Sign</u> Handshake Messages



Significant tail latencies for MLDSA leaf certificates

Measured Results: <u>Sign</u> Handshake Messages



Significant tail latencies for MLDSA leaf certificates

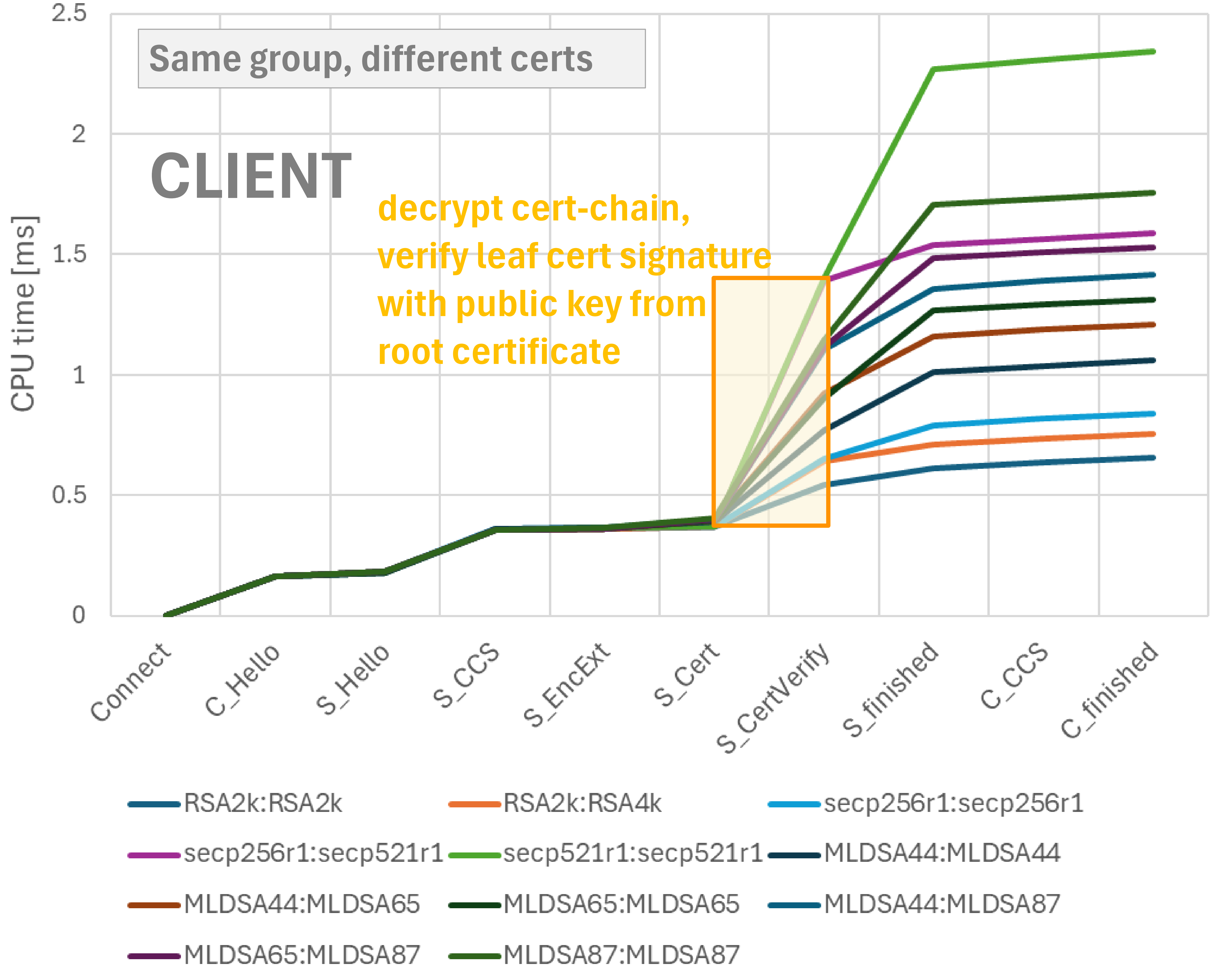
This is happening on the server-side – and the server will ALWAYS sign.

→ For a highly loaded server, ALL clients are affected by significant tail latencies

("highly loaded" = many parallel request, processed sequentially)

Measured Results: Verify Certificate(s)

Constant:
X25519MLKEM768
TLS_AES_128_GCM_SHA256
Root certificate sent but not verified

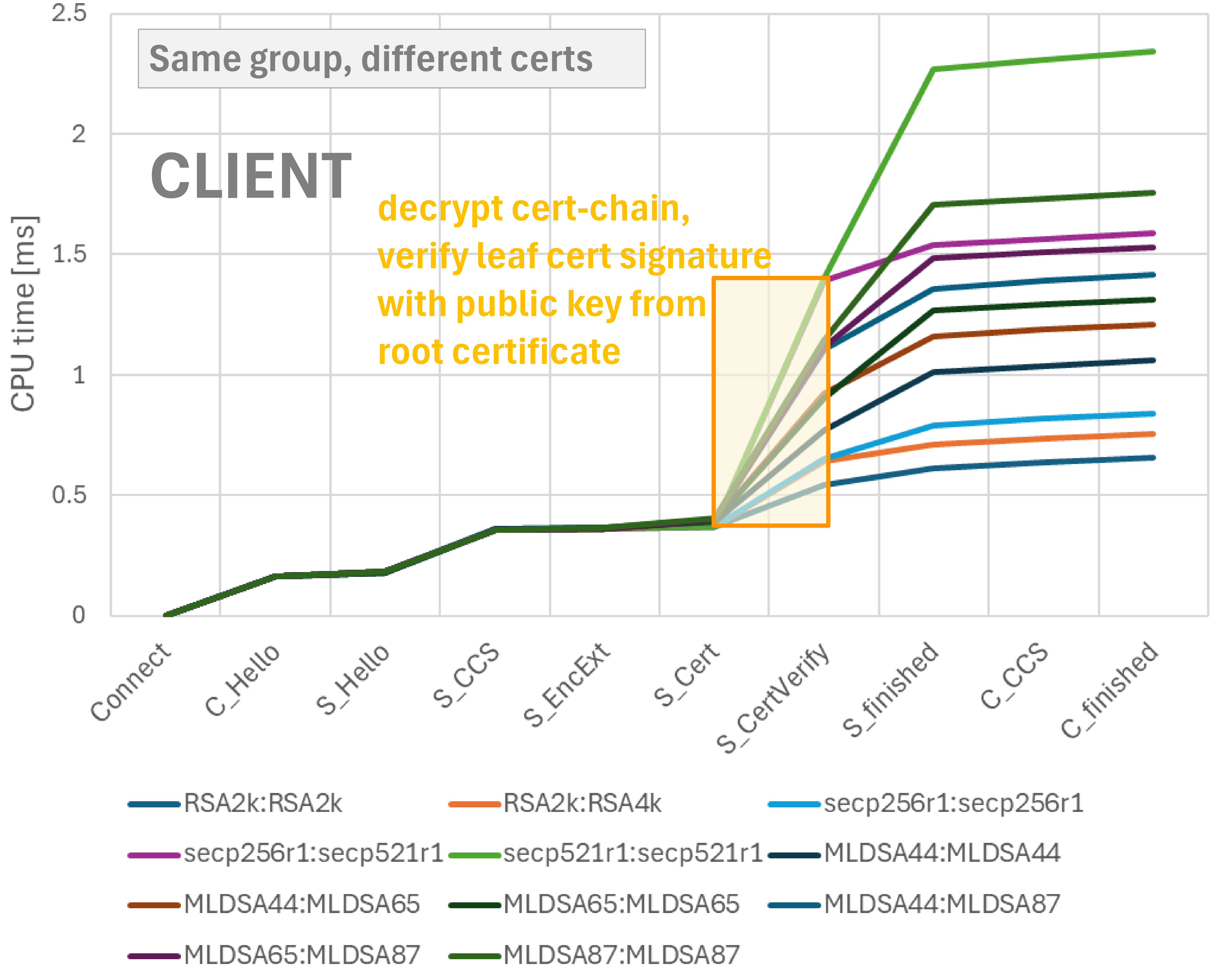


Certificate notation: <Leaf cert>:<Root cert>

The leaf certificate signature algorithm is not involved, only the algorithm(s) of the <u>parent cert(s)</u> are used.

Measured Results: Verify Certificate(s)

Constant:
X25519MLKEM768
TLS_AES_128_GCM_SHA256
Root certificate sent but not verified



Certificate notation: <Leaf cert>:<Root cert>

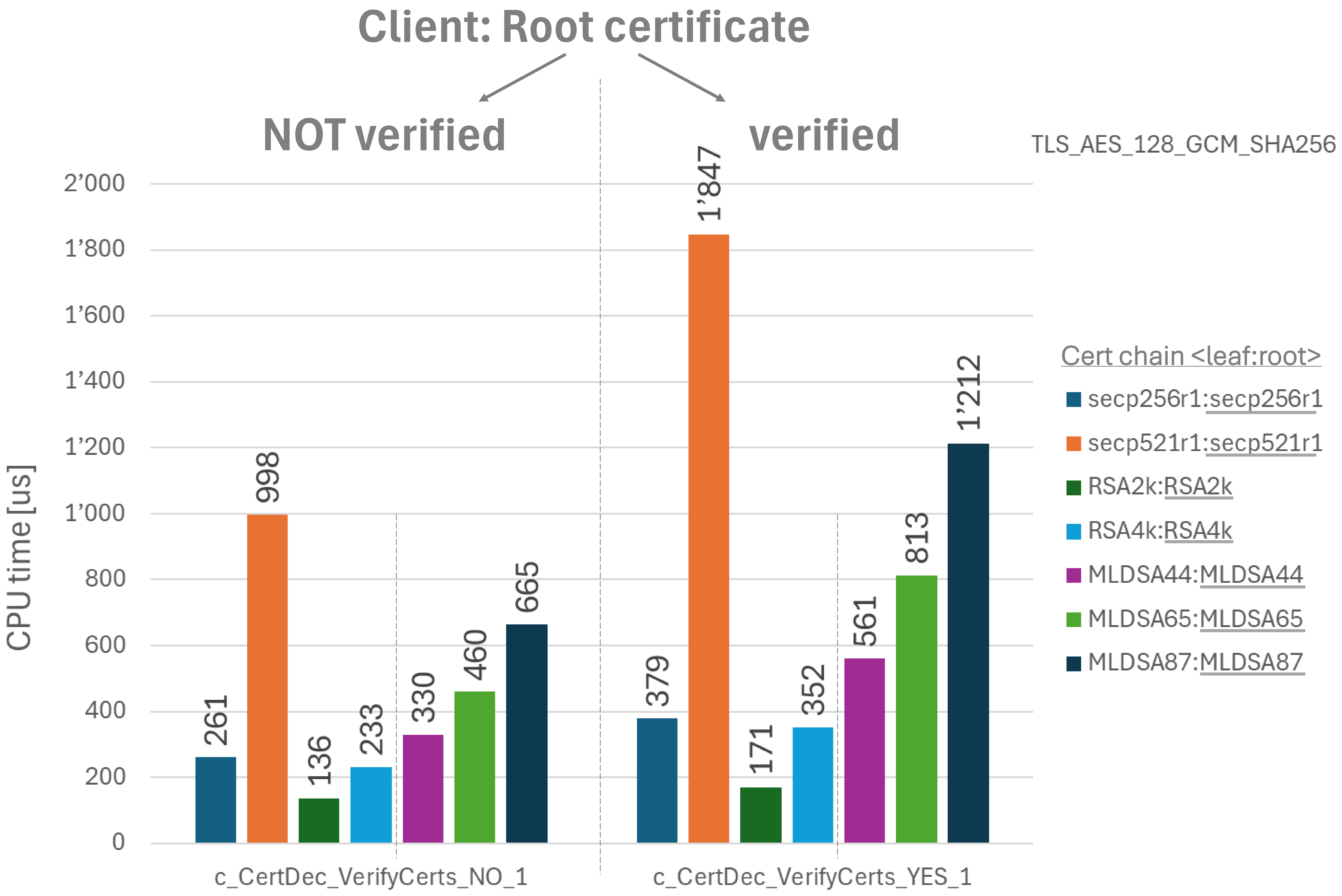
The leaf certificate signature algorithm is not involved, only the algorithm(s) of the parent cert(s) are used.

Variations:

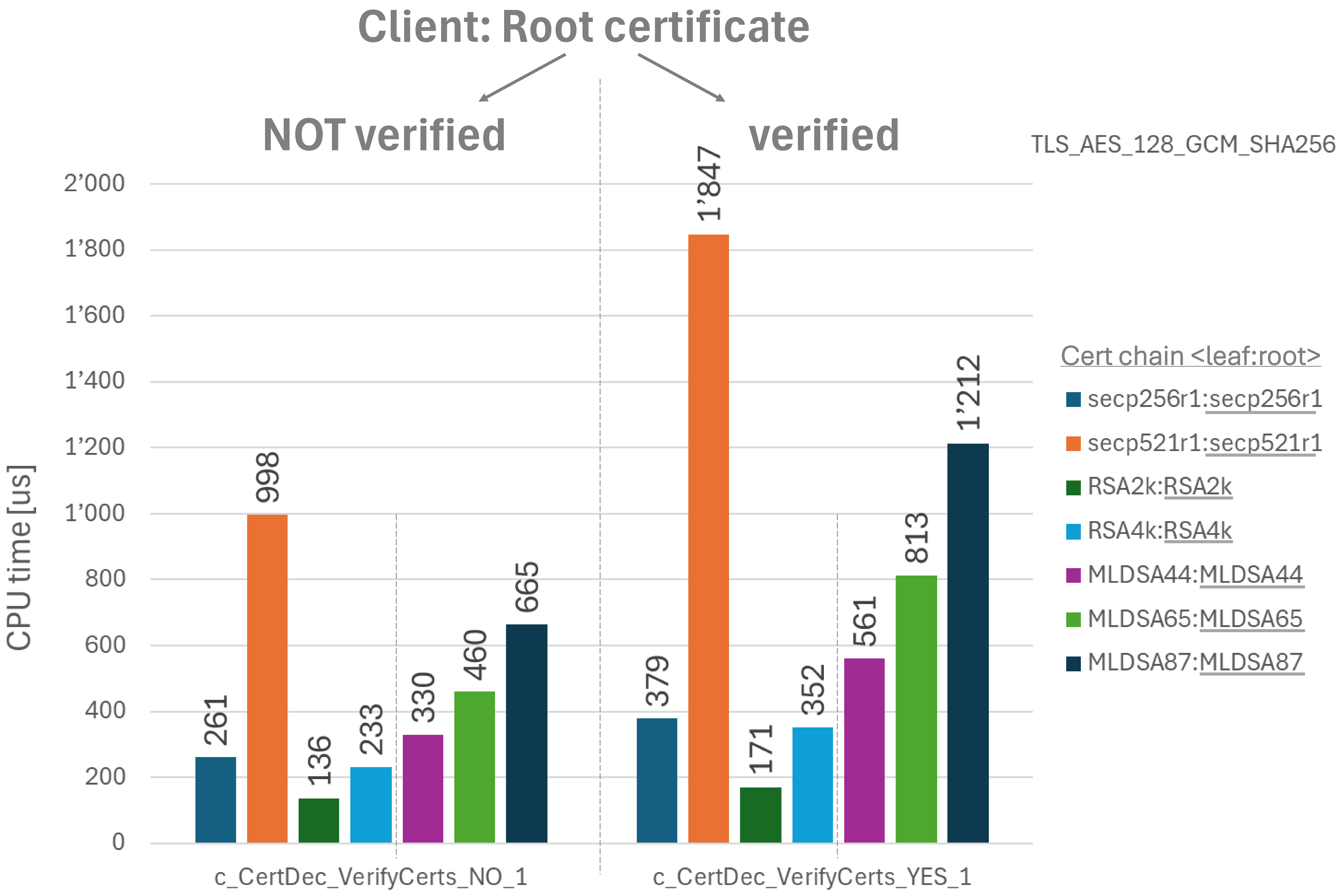
The root certificate can optionally be sent
Latency penalties from receiving, storing and decrypting an extra certificate is ~45 µs (± 15 µs pending cipher), which is small in the overall scheme.
Therefore, we only show results for the case where the root certificate is not sent and when using cipher TLS_AES_128_GCM_SHA256.

The root certificate can optionally be verified
This to capture the equivalent case of an intermediate certificate being in the chain.

Measured Results: Verify Certificate(s)



Measured Results: Verify Certificate(s)



All MLDSA versions beat a secp521r1 root|parent certificate

Having to verify a parent (e.g. intermediate) certificate virtually doubles the latencies

Latencies for Secp256r1 vs RSA4k are virtually the same

Using MLDSA[44|65|87] instead of P-256|RSA4k adds ~[100|200|400]µs

Overall Performance: Latencies

Parent cert NOT verified

group	cert_chain	15% quantile client	15% quantile server	15% quantile total	50% quantile client	50% quantile server	50% quantile total	99% quantile client	99% quantile server	99% quantile total
x25519	secp256r1:secp256r1	0.644	0.263	0.906	0.653	0.265	0.917	0.705	0.356	1.061
X25519MLKEM768	secp256r1:secp256r1	0.798	0.354	1.151	0.806	0.355	1.162	0.837	0.439	1.276
x25519	secp256r1:secp521r1	1.375	0.265	1.640	1.384	0.266	1.651	1.415	0.321	1.736
X25519MLKEM768	MLDSA65:MLDSA65	1.286	1.045	2.331	1.307	1.801	3.108	1.384	6.853	8.237
X25519MLKEM768	MLDSA65:MLDSA87	1.446	1.072	2.518	1.464	1.807	3.271	1.563	6.945	8.508
MLKEM1024	MLDSA87:MLDSA87	1.634	1.294	2.928	1.656	2.069	3.725	1.708	7.306	9.013
SecP384r1MLKEM1024	MLDSA87:MLDSA87	2.505	2.143	4.648	2.524	2.920	5.444	2.871	8.135	11.006

Parent cert verified

group	cert_chain	15% quantile client	15% quantile server	15% quantile total	50% quantile client	50% quantile server	50% quantile total	99% quantile client	99% quantile server	99% quantile total
x25519	secp256r1:secp256r1	0.764	0.265	1.029	0.781	0.266	1.047	0.825	0.357	1.182
X25519MLKEM768	secp256r1:secp256r1	0.905	0.353	1.258	0.918	0.355	1.274	0.973	0.375	1.348
x25519	secp256r1:secp521r1	2.200	0.264	2.464	2.216	0.265	2.481	2.330	0.325	2.655
X25519MLKEM768	MLDSA65:MLDSA65	1.635	1.057	2.693	1.657	1.808	3.465	1.840	6.932	8.772
X25519MLKEM768	MLDSA65:MLDSA87	1.975	1.074	3.049	1.997	1.814	3.811	2.824	6.995	9.819
MLKEM1024	MLDSA87:MLDSA87	2.161	1.295	3.456	2.186	2.001	4.187	2.253	7.303	9.556
SecP384r1MLKEM1024	MLDSA87:MLDSA87	3.031	2.154	5.185	3.062	2.929	5.992	3.353	8.148	11.501

OVERALL CPU TIMES (aka LATENCY):

Groups & Certificates vs
- Quantiles,
- Parent cert verification

Only some selected group & certificate chain combinations shown

Cipher : TLS_AES_128_GCM_SHA256

Timings in [ms]

Rules of thumb for latencies:

- **“Harvest now, decrypt later”**: Overall adds ~250 µs (~=25%) latency, server load goes up ~33%
  - To put things into perspective: Having a secp521r1 certificate in the chain hurts ~3x more than using X25519MLKEM768
- Using **MLDSA certificates** really hurts performance:
  - ~3x higher latency and server load in 50% quantile (compared to P-256),
  - up to ~10x higher latency and ~20x higher server load in 99% quantile

Overall Performance: Data volume vs Bandwidths

Relevance: new TLS session ≅ new client

Group	Cert chain	num certs	Data volume [Bytes/Session]		Max session rate vs data-center network BW [Msess/s]				Data transport latency not considering flight-time [us]			
			Client -> Server	Server -> Client	1 Gbps	10 Gbps	100 Gbps	400 Gbps	1 Gbps	10 Gbps	100 Gbps	400 Gbps
x25519	P-256:P-256	1	256	928	0.11	1.08	10.78	43.10	11.84	1.18	0.12	0.03
MLKEM512	P-256:P-256	1	1016	1665	0.06	0.60	6.01	24.02	26.81	2.68	0.27	0.07
MLKEM768	P-256:P-256	1	1400	1985	0.05	0.50	5.04	20.15	33.85	3.39	0.34	0.08
X25519MLKEM768	P-256:P-256	1	1432	2017	0.05	0.50	4.96	19.83	34.49	3.45	0.34	0.09
MLKEM1024	P-256:P-256	1	1784	2464	0.04	0.41	4.06	16.23	42.48	4.25	0.42	0.11
MLKEM512	MLDSA44:MLDSA44	1	1014	7612	0.01	0.13	1.31	5.25	86.26	8.63	0.86	0.22
MLKEM768	MLDSA65:MLDSA65	1	1398	10350	0.01	0.10	0.97	3.86	117.48	11.75	1.17	0.29
X25519MLKEM768	MLDSA65:MLDSA65	1	1430	10382	0.01	0.10	0.96	3.85	118.12	11.81	1.18	0.30
MLKEM1024	MLDSA87:MLDSA87	1	1782	14106	0.01	0.07	0.71	2.84	158.88	15.89	1.59	0.40
MLKEM512	MLDSA44:MLDSA65	2	1016	14060	0.01	0.07	0.71	2.84	150.76	15.08	1.51	0.38
MLKEM768	MLDSA65:MLDSA87	2	1400	19185	0.01	0.05	0.52	2.08	205.85	20.59	2.06	0.51
X25519MLKEM768	MLDSA65:MLDSA87	2	1432	19217	0.01	0.05	0.52	2.08	206.49	20.65	2.06	0.52
MLKEM1024	MLDSA87:MLDSA87	2	1782	21623	0.00	0.05	0.46	1.85	234.05	23.41	2.34	0.59

Latencies due to **data transport latencies are relatively small and hence of no big concern**: max +23 μs @ 10Gbps

Data transfers Server → Client are dominating

The data-center egress network BW becomes the limiting factor for TLS session establishment rates

Rules of thumb for session establishment rates:

- Protecting against **'harvest now, decrypt later'** lowers upper bound for session establishment capability by **factor of ~2**
- Adding a **2-stage Q-safe cert chain**, without sending the root certificate, cuts it by **~10x**
- When also sending the parent e.g. **intermediate certificate**, the cut is **~20x**

Insights Summary

CIPHERS

Choice of ciphers has a small impact on overall latency (variations are within ± 50 μs)

GROUPS

Protecting against '**harvest now, decrypt later**'

- "Pure" Q-safe key agreement algorithms are virtually at par with their legacy counterparts
- MLKEM512 (128 sec bits) is even faster than legacy groups
- Hybrid X25519MLKEM768 adds ~250 μs (~=25%) to overall latency, Server load goes up ~33%

CERTIFICATES

Using Q-safe algorithms for **authentication**

- MLDSA is costly per-se AND leads to unbound tail latencies:
  - ~3x higher latency in 50% quantile,
  - BUT up to ~10x higher latency and ~20x higher server load in 99% quantile
- Having to verify a parent (e.g. intermediate) certificate virtually doubles the latencies → use 2-stage cert chains and don't send the root certificate

DATA VOLUMES

Latencies due to **data transport latencies are relatively small and hence of no big concern**: max +23 μs @ 10Gbps
The data-center egress network BW becomes the limiting factor for TLS session establishment rates
Rules of thumb:

- Protecting against **'harvest now, decrypt later'** is cutting session establishment capability by **factor of ~2**
- Adding a **2-stage Q-safe cert chain**, without sending the root certificate, cuts session rates by **~10x**
- When also sending the parent e.g. **intermediate certificate**, the cut is **~20x**

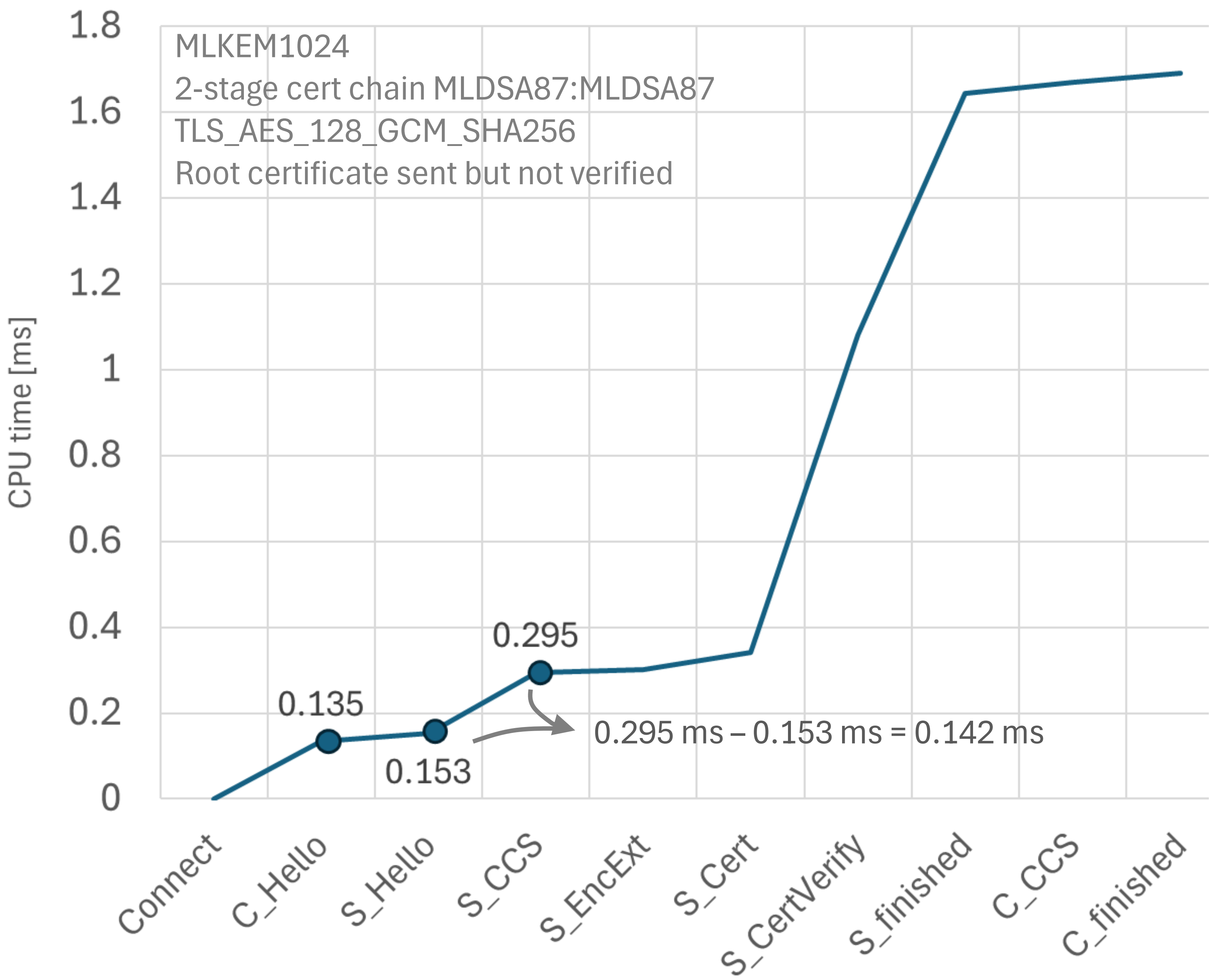
THANK YOU!

Backup

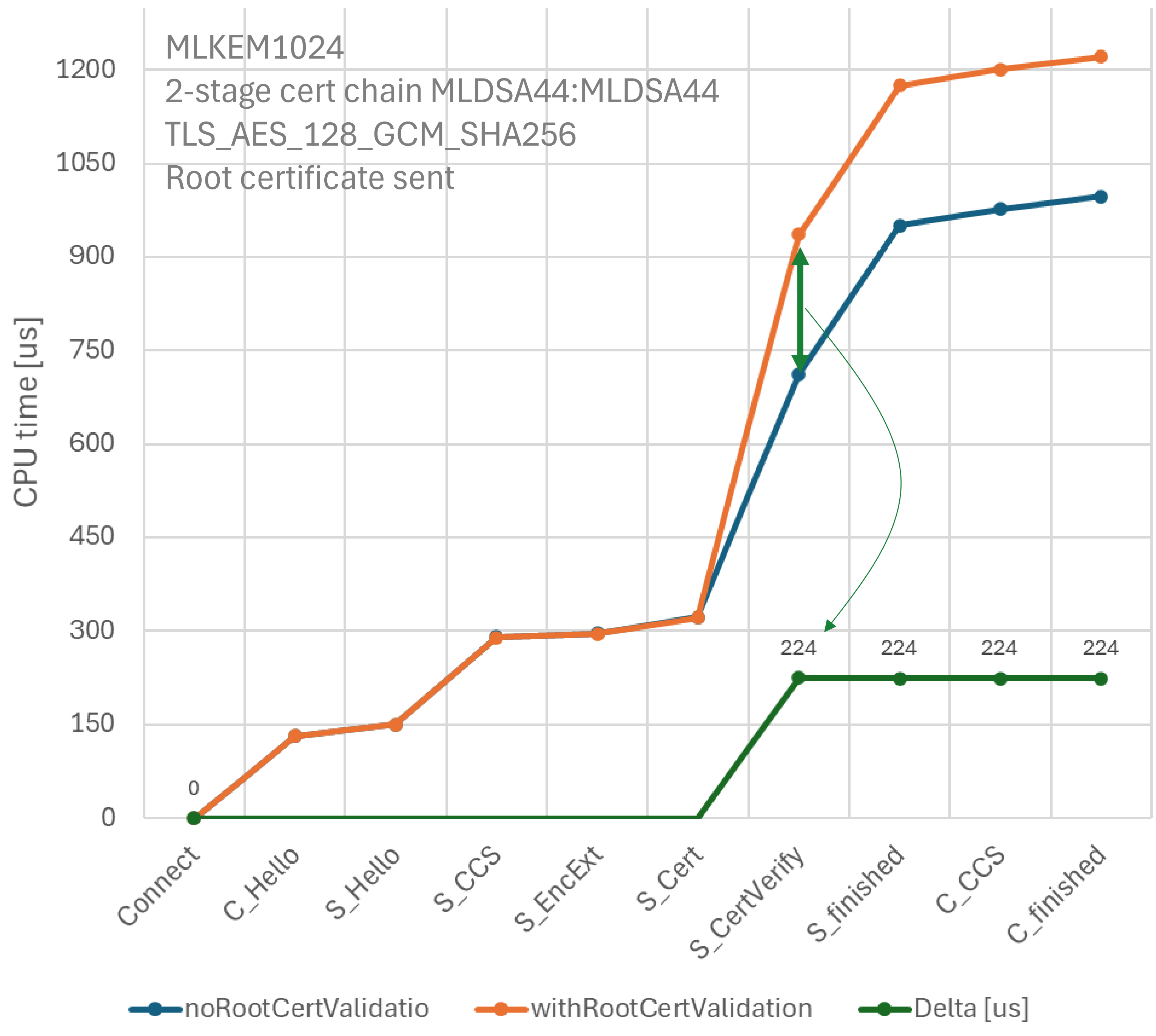
Data extraction: Examples (client-side)

Data extraction can be based on a *single measurement* OR based on the difference between two measurements

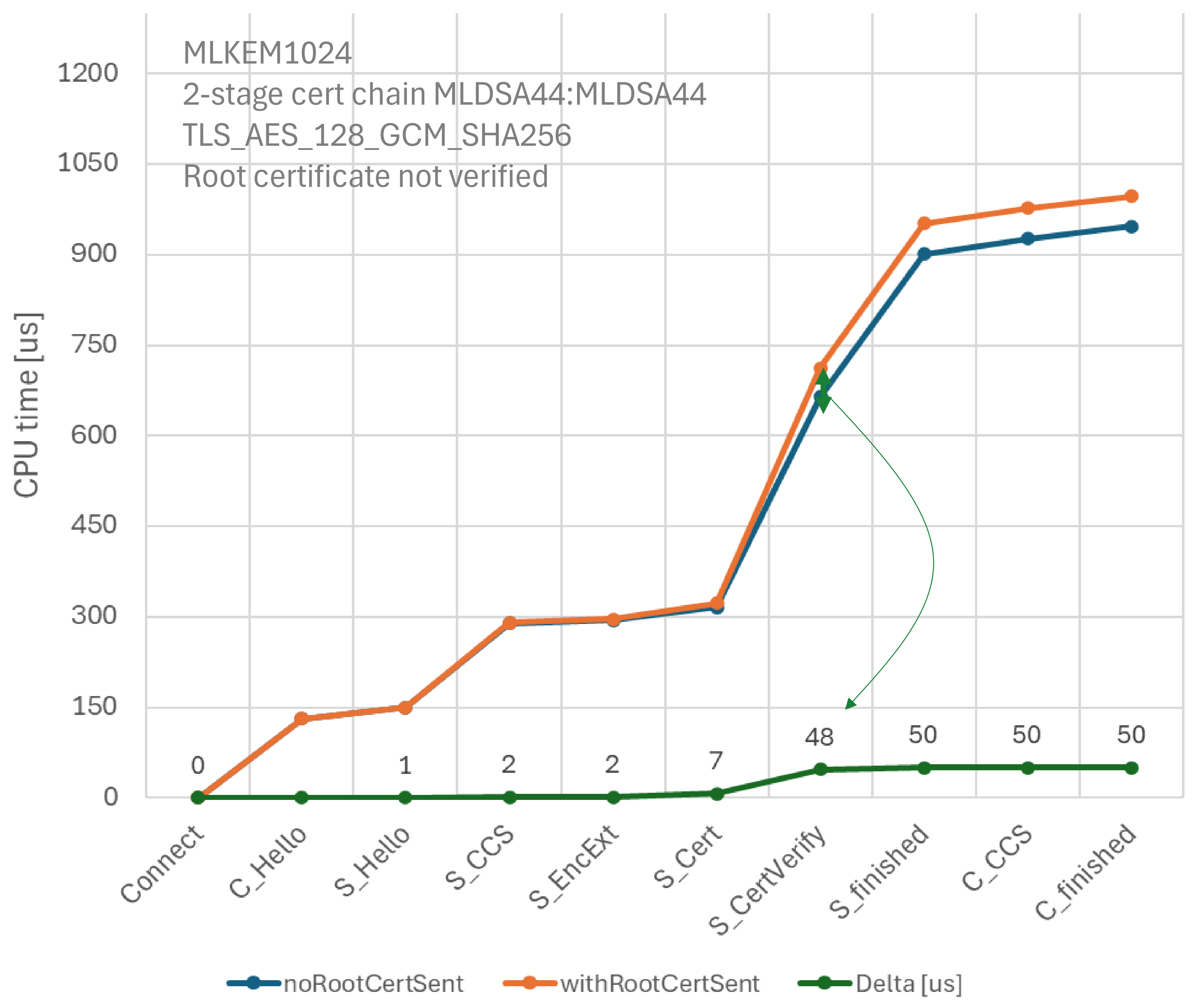
Key agreement



Impact of verifying the root certificate



Impact of sending the root certificate



Insight: It takes 135 µs for the client to generate a MLKEM1024 key pair and 142 µs to derive a common MLKEM1024 secret

Insight: It takes 224 µs for the client to verify the (self-)signature of a MLDSA44 root certificate

Insight: It takes 48 µs for the client to decrypt a MLDSA44 root certificate plus do the bookkeeping

Constant: CLIENT, MLKEM1024, TLS_AES_128_GCM_SHA256, MLDSA44:MLDSA44
Note: Generated with OpenSSL v3.5.0

Crypto-Algorithms: Available Options

orthogonal

Groups	Certificate algorithms [leaf:root]	Ciphersuites	Root certificate sent	Root certificate verified
MLKEM512	RSA2k:RSA2k	TLS_AES_128_GCM_SHA256	YES	YES
MLKEM768	RSA2k:RSA4k	TLS_AES_256_GCM_SHA384	NO	NO
MLKEM1024	secp256r1:secp256r1	TLS_CHACHA20_POLY1305_SHA256		
x25519	secp256r1:secp521r1	TLS_AES_128_CCM_SHA256		
secp256r1	secp521r1:secp521r1	TLS_AES_128_CCM_8_SHA256		
X25519MLKEM768	MLDSA44:MLDSA44			
SecP256r1MLKEM768	MLDSA44:MLDSA65			
x448	MLDSA65:MLDSA65			
secp384r1	MLDSA44:MLDSA87			
secp521r1	MLDSA65:MLDSA87			
	MLDSA87:MLDSA87			

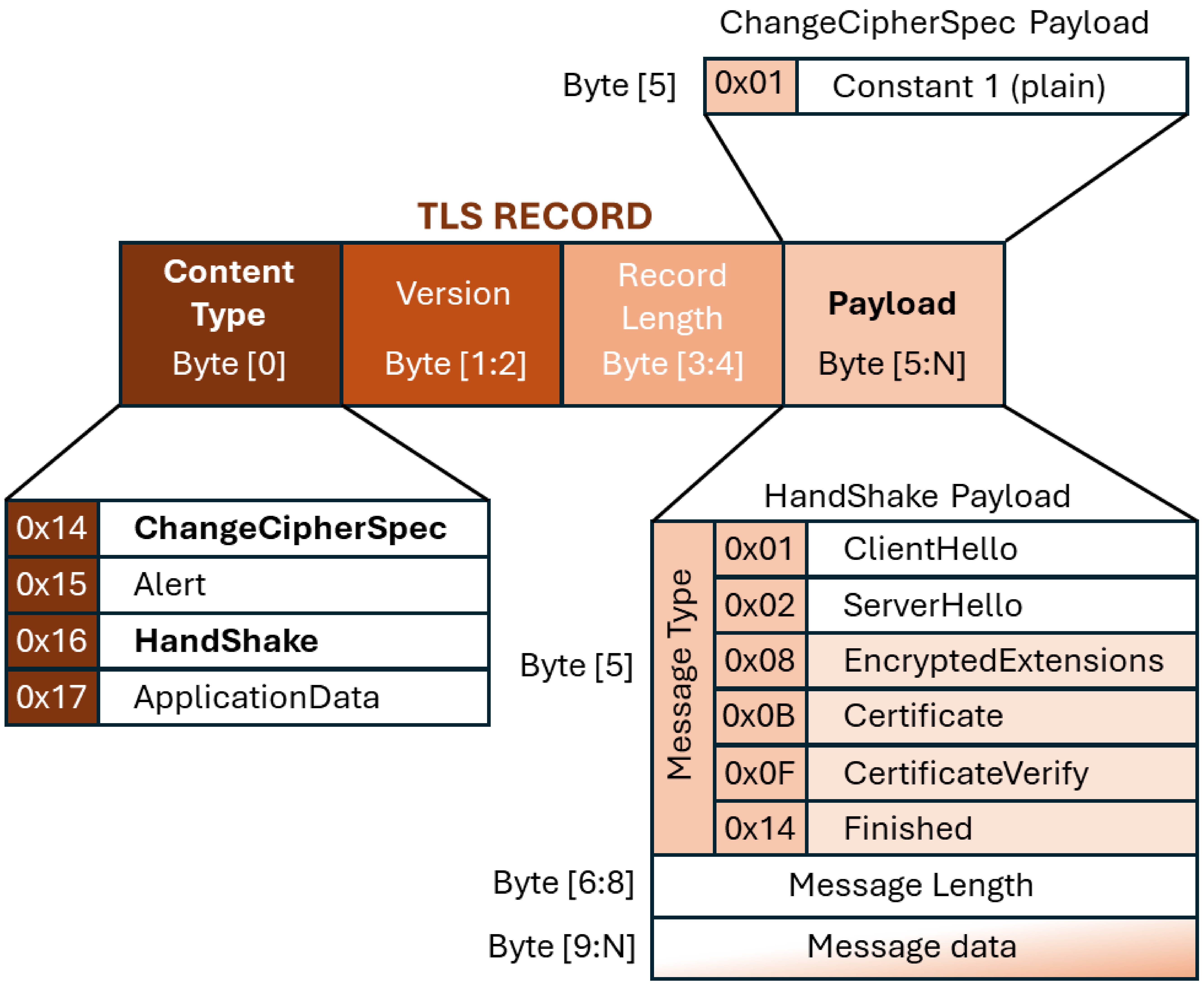
Derive duration to verify the signature on a certificate

Derive duration to encrypt or decrypt a certificate

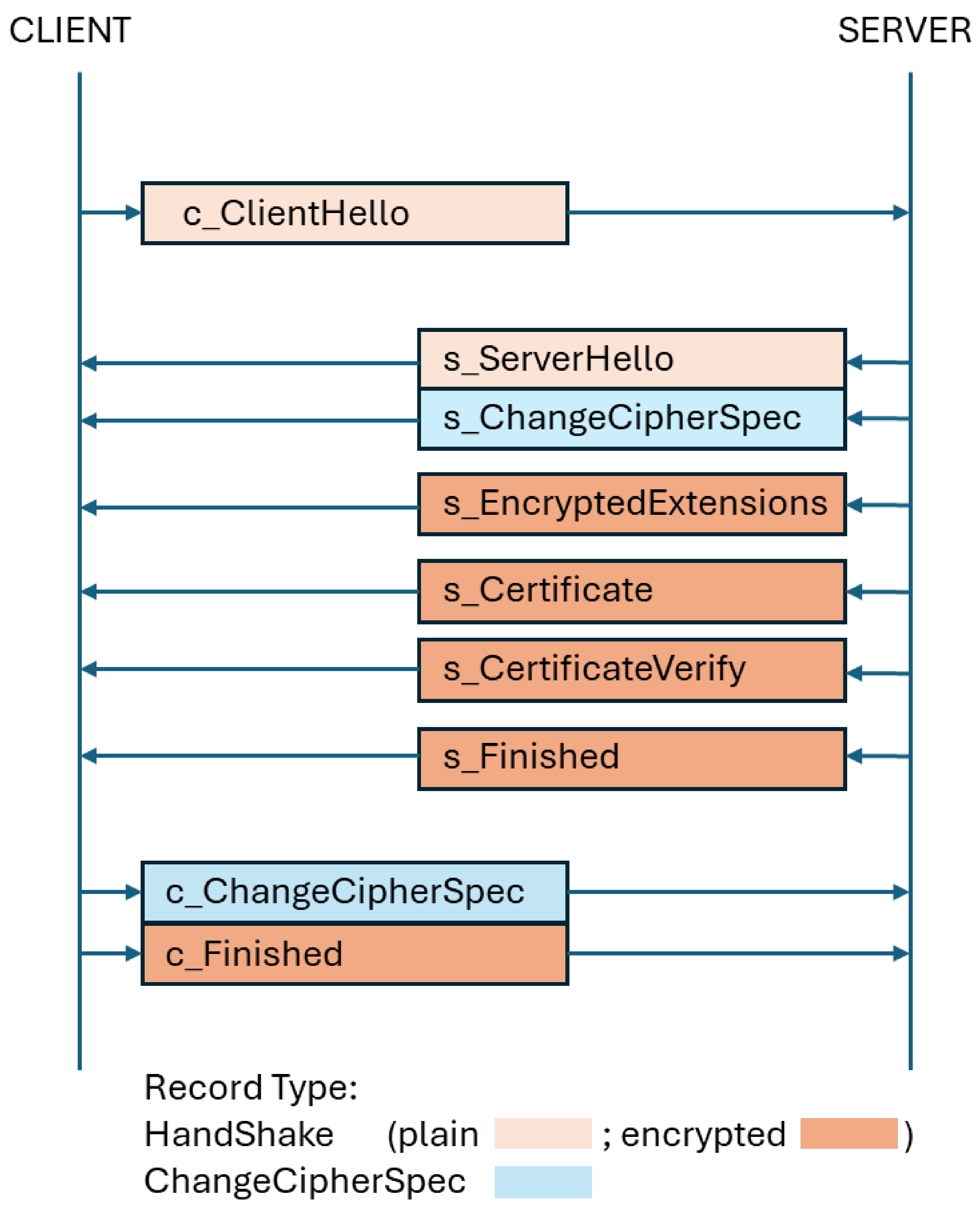
Derive duration to create a handshake digest and to sign or verify a signature

Derive duration for
- generating key-pair(s)
- generating common secret

Anatomy of a TLS v1.3 Record



TLS v1.3 session establishment sequence without HelloRetryRequest (HRR)



Note:
No HRR implies that the server can immediately accept a key-share offered by the client