# Security analysis of TLS composite ML-DSA authentication

This report in another language: 中文.

draft-ietf-tls-composite-mldsa-00 · ProVerif+ Agent report

| | |
|---|---|
| **Documents** | Use of Composite ML-DSA in TLS 1.3 (protocol definition); The Transport Layer Security (TLS) Protocol Version 1.3 (supporting); Post-Quantum Traditional (PQ/T) Hybrid Key Agreement Mechanisms for TLS 1.3 (supporting) |
| **Properties examined** | 12: 11 hold, 1 fail (1 attack) |
| **Findings** | None; known results reproduced in Appendix B: 2 |

**Contents**


1 Protocol and scope
2 Findings
3 Properties examined
4 Limits of this analysis
Appendix A: Reading of each property
Appendix B: Known results reproduced
Appendix C: Claims and prover answers
Appendix D: How the analysis was run


## 1 Protocol and scope

The draft puts composite ML-DSA into TLS 1.3 server authentication: the server certificate carries one ML-DSA key and one traditional key (Ed25519 or ECDSA), and the CertificateVerify carries a signature from each component over the same TLS signing input. The key agreement is X25519MLKEM768, whose shared secret concatenates an ML-KEM secret and an X25519 secret. A client that requires composite authentication holds the server's composite public key and checks both component signatures; a second client is also analysed that still accepts a traditional-only certificate.

When the exchange ends, the client is sure that the server holding the composite private key took part in this connection, that the signature was made with the negotiated scheme over this connection's transcript under the server context string, and that it holds traffic keys nobody else can derive. The server is sure the client computed the same shared secret and the same transcript. A leak of one component key leaves both assurances in place, and a leak of the whole long-term key after the session leaves the session's traffic keys secret because the key agreement is ephemeral.

The parties:

- **Client**: requires the composite certificate; pins the composite public key. Holds: pinned pk_MLDSA, pk_trad; trust in the certificate.
- **Server**: holds the composite long-term signing key. Holds: sk_MLDSA, sk_trad; certificate (key type composite, pk_MLDSA, pk_trad).

Figure 1 shows the exchange as the documents specify it.

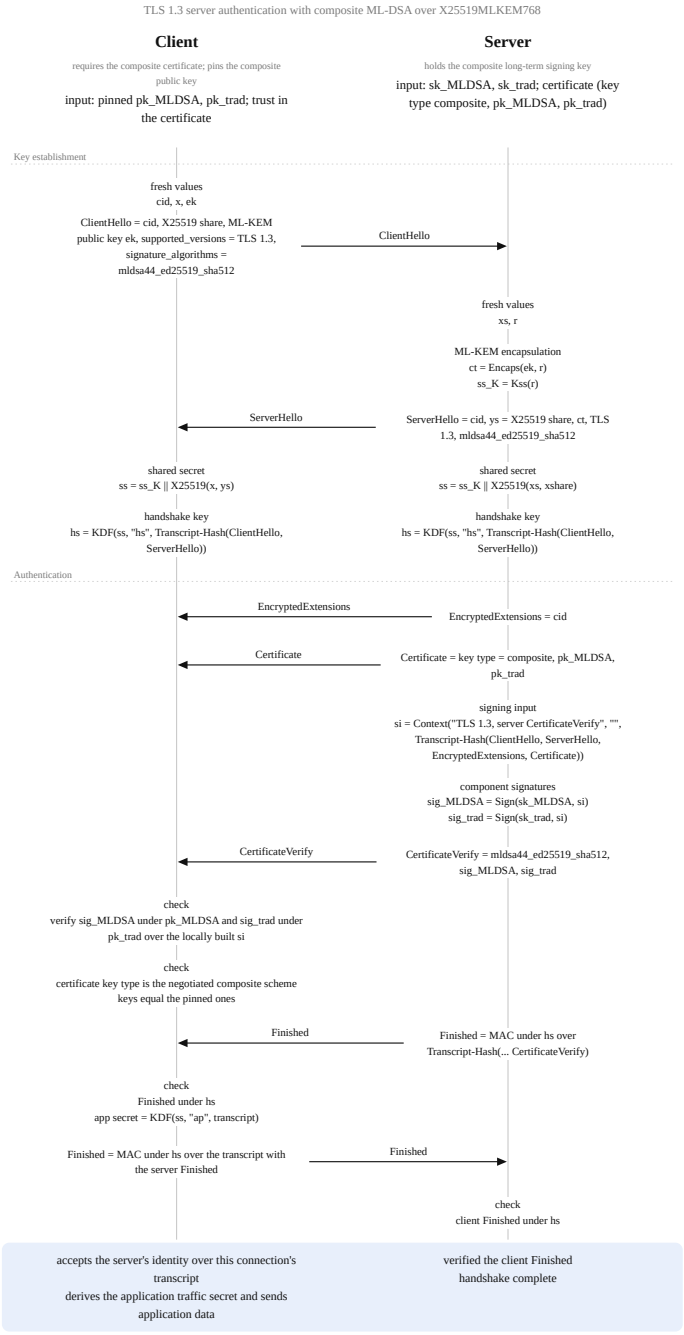



Figure 1. The protocol as specified.

The scope of the analysis is in Appendix D: what was analysed, what was left out, and the sections of the documents it is built from.

## 2 Findings

**No finding.** The analysis found nothing that the documents do not already state or set aside. Where a property fails, the documents know it: 2 such results are reproduced in Appendix B.

The composite handshake behaves as the draft intends within the analysed scope: against the network alone, with one component private key, with the X25519 half broken, with the session keys read and with a signature from another protocol context, the client accepts a server CertificateVerify only when the real server signed this connection's transcript under the composite scheme, and no restricted scheme, wrong key type or TLS 1.2 connection is accepted. The traffic secrets stay secret, and a long-term signing key that leaks after a session does not expose

them. The one failing property is the one the draft warns about: a client that also accepts a traditional-only certificate is impersonated with the traditional component key, while the composite client rejects the same attacker; it is reproduced here with the exact conditions, so it is a known result and not a finding. All twelve targets have an answer, but the proofs are proofs about the signature and the transcript: the handshake flight is unencrypted in the model and the key schedule is one-way, so nothing is shown about the record layer or about an attacker who learns the shared secret.

## 3 Properties examined

Each row is one property put to ProVerif. The result is ProVerif's answer on the analyst's model of the protocol; value and whether the documents already state the result are the analyst's judgement, with the reasons in Appendix A.

| ID | Property | Compromise assumed | Result | Value | In the documents |
|---|---|---|---|---|---|
| A02 | **Live server run behind every acceptance.** When the client accepts the server's CertificateVerify and completes the handshake, there exists a server run whose intended peer is that client. | no compromise | Holds | medium | yes |
| A03 | **Agreement on peers, scheme, secret, transcript.** When the client accepts server authentication, there exists a matching server run, and the two parties' identities, negotiated signature scheme, key-exchange shared secret and transcript agree. | no compromise | Holds | medium | yes |
| A04 | **One acceptance per server run.** Each time the client accepts server authentication, it corresponds to a distinct server run; the same server run cannot be accepted twice. | no compromise | Holds | medium | yes |
| A08 | **Scheme, key, context and transcript binding.** The signature scheme, certificate public key type, context string and transcript used by the CertificateVerify the client accepts all correspond to the client's proposal and the current connection context. | no compromise | Holds | medium | yes |
| K01 | **Secrecy of keys from the network.** The attacker cannot obtain the server's long-term signing private key, nor the handshake and traffic keys derived by an honest session. | no compromise | Holds | medium | yes |
| C01 | **Forward secrecy of the traffic keys.** Leaking the server's long-term signing private key after the session completes does not expose the handshake and traffic keys of previously completed sessions. | server's long-term signing private key (after the session completes) | Holds | low | yes |
| C08 | **Authentication survives one broken component.** When one of the traditional component signing private key or the ML-DSA component signing private key is compromised, the server authentication goal (the client accepts the real server) still holds. | one component private key of the composite signature (after the attacker obtains the component private key) | Holds | medium | yes |
| R05 | **No downgrade of scheme or version.** The signature scheme and protocol version the client accepts satisfy the minimum policy the draft specifies; the attacker cannot downgrade the accepted configuration by manipulating the negotiation. | no compromise | Holds | medium | yes |

| ID | Property | Compromise assumed | Result | Value | In the documents |
|---|---|---|---|---|---|
| C08.9 | **One leaked component keeps authentication.** When one component private key of the composite signature leaks, the client still accepts only the real server's authentication. | one component private key of the composite signature (after the attacker obtains the component private key) | Holds | medium | yes |
| A08.10 | **Signer role, context and transcript bound.** The signer role, context string and transcript of the CertificateVerify the client accepts all correspond to the real context of this connection. | no compromise | Holds | medium | yes |
| A08.11 | **Scheme compatible with certificate key.** The signature scheme the client accepts is compatible with the certificate public key type; on a mismatch the connection is rejected rather than accepted. | no compromise | Holds | medium | yes |
| X01 | **Traditional-only fallback impersonation.** A client that also accepts traditional-only server authentication accepts, against an attacker holding the traditional component private key, a server run that did not happen, while a client that requires the composite certificate keeps rejecting that attacker. | the private key of the traditional component (Ed25519/ECDSA) of the server's composite certificate, obtained once CRQCs are available, and can sign with that component only | Attack holds (stated in the documents) | medium | yes |

"Attack holds (stated in the documents)": the prover found the attack, and the documents themselves already state it. The property fails; this is a known result, not a finding (Appendix B).

Goals the documents state that the drafted list of properties did not have, added by the analyst (Appendix D): X01.

The questions asked, and the analyst's answers:

| Question | Answer | Rests on |
|---|---|---|
| Q1: Server authentication when a single signature component leaks | For a client that requires the composite certificate, yes: with exactly one component private key, and one component signature over the signing input of its choice, the attacker is never accepted, because the second component over the same transcript is missing; a client that also accepts traditional-only authentication is impersonated with that same single key. | C08, C08.9, A02, X01 |
| Q2: CertificateVerify role/context/transcript binding | The accepted CertificateVerify is bound to the server role and its context string, to the negotiated scheme and to this connection's transcript, and a composite signature made with the same key in another protocol context does not verify. | A08.10, A08, A02 |
| Q3: Signature scheme negotiation and rejection of certificate/public key algorithm mismatch | A mismatch is refused rather than accepted: no CertificateVerify naming an RSA-PKCS1-v1_5 composite scheme or stand-alone ML-DSA, and no certificate whose key type is not the negotiated composite scheme, is ever accepted. | A08.11, R05 |

## 4 Limits of this analysis

A result here is a result on a model of the protocol: it says what the specified messages and checks give or fail to give, with cryptography taken as ideal, and nothing about an implementation.

What this analysis does not cover, as the analyst states it:

- The handshake flight is modelled without its record-layer encryption, so nothing here rests on the AEAD protecting EncryptedExtensions, Certificate, CertificateVerify or Finished.
- The key schedule and the Finished MAC are one-way (private) functions, so the attacker cannot derive a traffic key even from a known shared secret; the secrecy answers are conservative in that direction and say nothing about an attacker who learns the shared secret itself.

- The attacker's one-component capability is a single component signature over one message, which is what one impersonation attempt needs, rather than a signing oracle for that component.
- X25519 is modelled with the single law of the shared secret plus a reduction that recovers the scalar, and ML-KEM as a symbolic KEM; no computational security and nothing about post-quantum security is shown.
- PKI is out of scope: the client pins the composite public key, and certificate chains, revocation and name binding are not modelled.
- HelloRetryRequest, PSK resumption, 0-RTT, client authentication, key update, the record layer, alerts and the enforcement of message ordering are not modelled.
- Any number of client and server sessions are considered, but no state is carried between sessions (no resumption secret, no ticket).

What stays open for each property, and what would settle it, is in Appendix A.

## Appendix A: Reading of each property

The analyst's own reading of the prover's answers, not an answer of the prover: what the answers mean for the protocol, what the reading rests on, whether it can be used as it stands, and what it is worth.

### A02: Live server run behind every acceptance

When a client accepts the server's CertificateVerify and completes the handshake, there exists a server run whose intended peer is that client.

The client accepts a CertificateVerify only when it holds the server's signatures over this connection's transcript, with the negotiated scheme and under the pinned certificate keys. This survives the leak of either single component key, of the session's traffic keys and of the X25519 scalars.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value medium: A proof of the draft's own requirement with the exact attackers named; the caveats bound what it is worth.
- Stated in the documents: yes. Where the documents say it: s0019, s0020, s0406
- Rests on: The handshake flight is not encrypted in the model, so the attacker reaches the certificate check freely and the proof rests on the composite signature rather than on the record layer.; The key schedule is one-way, so the attacker cannot compute the Finished values even when it plays a server.; The client pins the composite public key; no certificate chain is modelled.
- Open: The attacker cannot be driven through the Finished check in this model, so the client's completion is not shown to be protected by the same argument.

### A03: Agreement on peers, scheme, secret, transcript

When the client accepts the server's authentication, there exists a matching server run, and the two sides agree on the identities, the negotiated signature scheme, the key-exchange shared secret and the transcript.

When the client completes, the server sent its Finished for the same fresh transcript value, the same concatenated X25519MLKEM768 secret and the same handshake traffic secret, so the two sides agree on the scheme and the keys.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value medium: Shows the transcript and the key agreement are tied together as the draft relies on.
- Stated in the documents: yes. Where the documents say it: s0357, s0410, s0561
- Rests on: The shared secret is the concatenation of the two component secrets, with X25519 as a plain key agreement.; Only the client's completion is related to the server's Finished; the server's own completion after the client Finished is outside this claim.

### A04: One acceptance per server run

Every time the client accepts the server's authentication, it corresponds to a distinct server run; the same server run cannot be accepted twice.

Each acceptance corresponds to a distinct server CertificateVerify for that same fresh transcript, including when the attacker can produce a different valid signature over an already signed transcript. A signature made for one connection is not accepted in another, because the transcript carries the client's fresh value.

- Result: Holds; answers what is asked: yes; usable
- Value medium: It is exactly the argument the draft makes for why losing SUF-CMA costs nothing in TLS, now shown mechanically.
- Stated in the documents: yes. Where the documents say it: s0046

- Rests on: The client's fresh value is a fresh name per client run, and the transcript hash is taken over it and both key shares.

### A08: Scheme, key, context and transcript binding

The signature scheme, certificate public key type, context string and transcript used by the CertificateVerify the client accepts all correspond to the client's offer and the current connection context.

The scheme the client accepts is the scheme the server signed and the client offered; no RSA-PKCS1-v1_5 composite scheme, no stand-alone ML-DSA and no TLS 1.2 connection is ever accepted. A composite signature made with the same key in another protocol context does not verify.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value medium: The bindings hold exactly, but the policy that rules out the stand-alone fallback is an assumption of the model, not something the draft fixes.
- Stated in the documents: yes. Where the documents say it: s0017, s0023, s0025, s0026, s0417, s0423, s0425
- Rests on: The client's policy is fixed by the model: it offers only the composite scheme, so the stand-alone ML-DSA fallback is answered for a relying party that has already chosen that policy.; The other protocol is modelled as signing an attacker-chosen message with the same key under a different context string.
- Open: The model's client offers only TLS 1.3, so a TLS 1.2 connection is refused by construction rather than by a separate check on a received message.

### A08.10: Signer role, context and transcript bound

The signer role, context string and transcript of the CertificateVerify the client accepts all correspond to the real context of this connection.

The CertificateVerify the client accepts covers this connection's transcript, built from the client's fresh value and both key shares, under the server context string. A composite signature the key holder made in another protocol context never verifies as a TLS CertificateVerify.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value medium: Shows mechanically that the empty composite context string is safe in TLS only because the TLS context string and the transcript are inside the signed input.
- Stated in the documents: yes. Where the documents say it: s0017, s0410, s0417
- Rests on: Other-context signatures are made over the same message with a different context string, so the separation rests entirely on the context string and the transcript, which is what the draft says it rests on.

### A08.11: Scheme compatible with certificate key

The signature scheme the client accepts is compatible with the certificate public key type; on a mismatch the connection is rejected rather than accepted.

The certificate the client accepts has exactly the key type of the negotiated composite scheme; a certificate of another key type is never accepted, so a scheme and a key cannot be paired wrongly.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value medium: Confirms the check the draft requires is present and not bypassable in this flow.
- Stated in the documents: yes. Where the documents say it: s0020, s0387, s0425
- Rests on: The key type is a value the model's certificate carries; the real binding is the certificate encoding, which the model represents by a constant.

### K01: Secrecy of keys from the network

The attacker cannot obtain the server's long-term signing private key, nor the handshake and traffic keys derived by an honest session.

Neither the handshake traffic secret nor the application traffic secret of an honest session is derivable by a network attacker, and the whole composite long-term key is not obtained during a session. The traffic secrets stay out of reach even when a CRQC recovers the X25519 scalars, because the ML-KEM secret is still needed.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value medium: The X25519-break half is the interesting one: the hybrid keeps the keys secret when one component falls.
- Stated in the documents: yes. Where the documents say it: s0406, s0489, s0561
- Rests on: The key schedule is one-way in the model, so this proof is conservative and says nothing about a shared-secret leak.; The ML-KEM and X25519 components are symbolic; no computational or quantum security is claimed.
- Open: A leak of the shared secret itself is not asked, so nothing is said about an attacker who learns it.

### C01: Forward secrecy of the traffic keys

After the session completes, a leak of the server's long-term signing private key does not expose the handshake and traffic keys of previously completed sessions.

A server long-term signing key that leaks after the session does not expose that session's application traffic secret, because the key agreement is ephemeral and the long-term key is not one of its inputs. Past traffic stays confidential once the composite signing key is broken.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value low: It follows directly from the design, since the signing key is not an input to the key schedule; the draft already plans the migration on this basis.
- Stated in the documents: yes. Where the documents say it: s0045, s0047
- Rests on: The leak is modelled after the session; sessions starting afterwards are outside this claim.; The key schedule is one-way in the model.
- Open: The handshake traffic secret after the leak is not asked; only the application traffic secret is.

### C08: Authentication survives one broken component

If one of the traditional component signing private key or the ML-DSA component signing private key is compromised, the server authentication goal (the client accepts the real server) still holds.

With exactly one component private key broken, whichever of the two, the client still accepts a server CertificateVerify only when the server signed this connection's transcript. One broken component does not break TLS server authentication, which is the draft's defense-in-depth claim.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value medium: It is the draft's central security claim, now shown mechanically with the exact attacker; the caveats on the attacker's power and on the key schedule keep it short of a full proof.
- Stated in the documents: yes. Where the documents say it: s0045, s0046, s0048
- Rests on: The broken-component attacker obtains one component signature over the signing input of its choice, which is what a single impersonation attempt needs; it is not given an unbounded signing oracle for that component.; The handshake flight is not encrypted, so the attacker does reach the certificate check.; The key schedule is one-way, so the attacker cannot complete the handshake even with a valid CertificateVerify.
- Open: A statement about a signing oracle for the broken component, rather than one signature, is not asked.

### C08.9: One leaked component keeps authentication

When one component private key of the composite signature leaks, the client still accepts only the real server's authentication.

The same conclusion holds when it is the component key that has leaked during the session rather than before it: the client still accepts only a CertificateVerify the server produced for that transcript.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value medium: States the in-session form of the leak the draft discusses, with the same conditions as C08.
- Stated in the documents: yes. Where the documents say it: s0045, s0048
- Rests on: Same considerations as C08: one component signature over one message, unencrypted flight, one-way key schedule.

### R05: No downgrade of scheme or version

The signature schemes and protocol versions the client accepts satisfy the minimum policy set by the draft; the attacker cannot downgrade the accepted configuration by manipulating the negotiation.

The client's checks on scheme, certificate key type and protocol version are all in place: a CertificateVerify naming a restricted scheme, a certificate of another key type or a TLS 1.2 connection is refused rather than accepted. The stand-alone ML-DSA fallback is excluded only for the policy the model assumes.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value medium: The refusal checks are confirmed, but the downgrade answer is only as good as the policy the model fixes.
- Stated in the documents: yes. Where the documents say it: s0023, s0025, s0026, s0052
- Rests on: The policy that refuses stand-alone ML-DSA is a choice of the model, as the draft says the matter is one of relying-party policy.; The version is checked on the ServerHello; the client offers only TLS 1.3.
- Open: The model fixes the relying-party policy to composite-only; a deployment with another policy is not analysed.

### X01: Traditional-only fallback impersonation

A client that also accepts traditional-only server authentication accepts, against an attacker holding the traditional component private key, a server run that did not happen, while a client that requires the composite certificate keeps rejecting that attacker.

A client that also accepts a traditional-only server certificate is impersonated by an attacker who holds only the traditional component key: it signs this connection's signing input, presents a traditional certificate, and is accepted with no server run at all. The client that requires the composite certificate keeps rejecting the same attacker, so the difference is the certificate key type the client is willing to accept.

- Result: Attack holds (stated in the documents); answers what is asked: yes; usable with caveats
- Value medium: The draft states the exposure; the model reproduces it with the exact conditions and the exact difference from the composite client.
- Stated in the documents: yes. Where the documents say it: main:558, s0048, s0052
- Rests on: The traditional-only client is modelled with its certificate check in the clear and no Finished exchange; the claim is about the certificate key type and the component signature, not about the record layer.; The attacker obtains one traditional component signature over the signing input of its choice, which is what holding that component key gives.
- Open: What a client that caches a previous post-quantum capability (the continuity mechanism the draft points to) would do is not analysed.

# Appendix B: Known results reproduced

A property that fails as the documents already state, or that they set aside, is not a finding. What follows is of that kind: the analysis reproduces it on the model, which gives the exact conditions.

## B.1 A client that also accepts traditional-only certificates is impersonated with the traditional component key

| | |
|---|---|
| **Kind** | Weakness of the protocol as specified |
| **Stated in the documents** | main:558 |
| **Value** | medium |
| **Properties** | X01 (section 3) |
| **Source text** | s0048, s0052 |
| **Shown by** | ProVerif attack query `x01_fallback` |

**What happens.** A client that accepts a traditional-only server certificate is impersonated by an attacker who holds only the traditional component key. The attacker obtains a traditional component signature over the TLS signing input built from that client's own fresh value, sends a certificate whose key type is traditional only, and the client accepts. In the same model the client that requires the composite certificate keeps rejecting the same attacker.

**What the attacker needs.** The traditional component private key of the server (available to an attacker with a CRQC, s0048) and a client that also accepts traditional-only server authentication.

**What the attacker gains.** The attacker becomes that server for that client: it is accepted, derives the keys the client derives, and the client believes it reached the server named in the traditional certificate.

**Proposed change.** Remove the weaker acceptance path: the client must offer only the composite signature scheme, require the certificate key type to be the negotiated composite key type, and verify both component signatures over the TLS signing input. That change was put into a copy of the model and checked: the impersonation is gone, the normal run is still found, and all of the other recorded answers are unchanged.

**Check of the change with ProVerif.** The change was put into a copy of the model: The client that used to accept traditional-only server authentication no longer does: it offers only the composite signature scheme, requires the certificate key type to be the negotiated composite key type, and verifies BOTH component signatures over the TLS signing input. Nothing else changes. On the changed model ProVerif proves `x01_fallback`: the attack is no longer possible there. The normal run of the protocol is still found. The other 17 recorded answers are unchanged. This is a first check on the model: it shows that the change removes this attack there, not that it introduces no other problem. Changed model: `records/fixes/0072-fix-fallback.pv`.

## B.2 Composite verification keeps server authentication when one component falls

| | |
|---|---|
| **Kind** | Guarantee confirmed |
| **Stated in the documents** | main:527 |
| **Value** | medium |
| **Properties** | C08, C08.9, A02 (section 3) |
| **Source text** | s0045, s0046 |

**What happens.** With exactly one component private key broken (traditional or ML-DSA), and with one component signature over the signing input of its own choice at its disposal, the attacker never makes the client accept a CertificateVerify the server did not sign for that connection. What it lacks is the second component signature over the

same transcript, and it cannot move one there from another connection because the transcript carries the client's fresh value.

## Appendix C: Claims and prover answers

The prover's record for each target: the level reached, the query that reached it, whether a control was checked, and the attacks with the analyst's verdict on each.

| target | level | bound | lemma | control | attacks | claims |
|---|---|---|---|---|---|---|
| A02 | proved | | a02_agree | checked | | 6 proved |
| A03 | proved | | a03_agree | checked | | 1 proved |
| A04 | proved | | a04_inj | checked | | 2 proved |
| A08 | proved | | a02_agree | checked | | 4 proved |
| K01 | proved | | k01_hs | checked | | 6 proved |
| C01 | proved | | c01_fs | checked | | 1 proved |
| C08 | proved | | a02_agree | checked | | 3 proved |
| R05 | proved | | a08_pkcs15 | checked | | 3 proved |
| C08.9 | proved | | a02_agree | checked | | 2 proved |
| A08.10 | proved | | a02_agree | checked | | 2 proved |
| A08.11 | proved | | a08_keytype | checked | | 1 proved |
| X01 | none | | | | x01_fallback (holds_in_source) | 1 attack |

Claim by claim, in the analyst's wording of each claim; the answer to each is the prover's:

| target | claim | attacker | answer | query | control |
|---|---|---|---|---|---|
| A02 | Every CertificateVerify the client accepts was signed by the server over this connection's transcript with the negotiated composite scheme and the pinned certificate key type; the attacker has the network only. | `network` | proved | a02_agree | checked |
| A02 | The client still accepts only a CertificateVerify the server signed, when the attacker also holds the traditional component private key of the composite certificate and can sign with that component. | `comp_traditional` | proved | a02_agree | checked |
| A02 | The client still accepts only a CertificateVerify the server signed, when the attacker also holds the ML-DSA component private key and can sign with that component. | `comp_mldsa` | proved | a02_agree | checked |
| A02 | The client still accepts only a CertificateVerify the server signed, when the attacker can also produce a different valid composite signature over a transcript the server has already signed. | `suf_cma` | proved | a02_agree | checked |
| A02 | The client still accepts only a CertificateVerify the server signed, when the attacker also recovers the X25519 ephemeral scalars with a CRQC. | `crqc_x25519` | proved | a02_agree | checked |
| A02 | The client still accepts only a CertificateVerify the server signed, when the attacker also reads the session's handshake and application traffic secrets. | `session_state` | proved | a02_agree | checked |
| A03 | When the client completes the handshake, the server sent its Finished for the same fresh transcript value, the same X25519MLKEM768 shared secret and the same handshake traffic secret; the attacker has the network only. | `network` | proved | a03_agree | checked |

| target | claim | attacker | answer | query | control |
|---|---|---|---|---|---|
| A04 | Each acceptance of a CertificateVerify by the client corresponds to a distinct CertificateVerify of the server for that same fresh transcript, so a signature made for one connection is not accepted in another; network only. | `network` | proved | a04_inj | checked |
| A04 | Still one acceptance per server run when the attacker can produce a different valid signature over an already signed transcript, because the transcript carries a fresh client value per connection. | `suf_cma` | proved | a04_inj | checked |
| A08 | No CertificateVerify naming an RSA-PKCS1-v1_5 composite scheme is ever accepted by the client. | `network` | proved | a08_pkcs15 | checked |
| A08 | No CertificateVerify naming stand-alone ML-DSA is ever accepted by a client that requires the composite scheme. | `network` | proved | a08_sa | checked |
| A08 | The signature scheme named in the CertificateVerify the client accepts is the one the server signed with and the one the client offered; network only. | `network` | proved | a02_agree | checked |
| A08 | A composite signature made with the same key in another protocol context is never accepted: the accepted CertificateVerify carries the server CertificateVerify context string and this connection's transcript. | `cross_protocol` | proved | a02_agree | checked |
| K01 | The handshake traffic secret of an honest session is not derivable by the attacker; network only. | `network` | proved | k01_hs | checked |
| K01 | The application traffic secret of an honest session is not derivable by the attacker, although it is used to protect a data record on the wire; network only. | `network` | proved | k01_app | checked |
| K01 | The attacker does not obtain the complete composite long-term signing key (both component private keys) during the session; network only. | `network` | proved | k01_ltk | checked |
| K01 | The handshake traffic secret stays out of reach when the attacker recovers the X25519 ephemeral scalars, because the X25519MLKEM768 secret also contains the ML-KEM secret. | `crqc_x25519` | proved | k01_hs | checked |
| K01 | The application traffic secret stays out of reach when the attacker recovers the X25519 ephemeral scalars, because the ML-KEM half of the shared secret is still needed. | `crqc_x25519` | proved | k01_app | checked |
| K01 | The complete composite long-term signing key is still not obtained when the attacker recovers the X25519 ephemeral scalars. | `crqc_x25519` | proved | k01_ltk | checked |
| C01 | After the session has completed and both component private keys of the server have leaked, the application traffic secret of that session is still not derivable by the attacker. | `ltk_after_session` | proved | c01_fs | checked |
| C08 | With exactly one component private key (the traditional one) and signing power with it, the client still accepts only a CertificateVerify the server produced for that transcript. | `comp_traditional` | proved | a02_agree | checked |
| C08 | With exactly one component private key (the ML-DSA one) and signing power with it, the client still accepts only a CertificateVerify the server produced for that transcript. | `comp_mldsa` | proved | a02_agree | checked |

| target | claim | attacker | answer | query | control |
|---|---|---|---|---|---|
| C08 | Reading the session's handshake and application traffic secrets does not let the attacker produce an acceptable CertificateVerify. | `session_state` | proved | a02_agree | checked |
| R05 | No CertificateVerify naming an RSA-PKCS1-v1_5 composite scheme is accepted, so the draft's restriction of those schemes to the certificate extension holds. | `network` | proved | a08_pkcs15 | checked |
| R05 | A client whose policy requires composite authentication never falls back to stand-alone ML-DSA when the attacker manipulates the negotiation. | `network` | proved | a08_sa | checked |
| R05 | No CertificateVerify of this document's composite schemes is accepted in a TLS 1.2 connection: the client only goes on when the selected version is TLS 1.3. | `network` | proved | a08_ver12 | checked |
| C08.9 | A leaked traditional component private key during the session still leaves the client accepting only a CertificateVerify the server produced for that transcript. | `comp_traditional` | proved | a02_agree | checked |
| C08.9 | A leaked ML-DSA component private key during the session still leaves the client accepting only a CertificateVerify the server produced for that transcript. | `comp_mldsa` | proved | a02_agree | checked |
| A08.10 | The CertificateVerify the client accepts covers the transcript hash of this connection, built from the client's fresh value and both key shares; network only. | `network` | proved | a02_agree | checked |
| A08.10 | A composite signature the holder of the server key made in another protocol context does not verify as a TLS CertificateVerify, because the signing input carries the server CertificateVerify context string and this connection's transcript. | `cross_protocol` | proved | a02_agree | checked |
| A08.11 | The key type of the certificate the client accepts equals the negotiated composite scheme; a certificate of another key type is never accepted. | `network` | proved | a08_keytype | checked |
| X01 | A client that also accepts traditional-only server authentication accepts a server run that never happened when the attacker holds the traditional component private key. | `comp_traditional` | attack (holds_in_source) | x01_fallback (the control of a02_agree) | |

## Appendix D: How the analysis was run

An LLM analyst builds the model of the protocol from the documents and puts each property to ProVerif; every result in section 3 is an answer of the prover.

ProVerif+ Agent session session-20261114T230251, 2026-11-14. Status: finished. Model: deepseek-v4-1-flash-261017. Steps: 74, prover 78 s, wall 22 min, cost 0.201 USD.

Context: the host removed stale parts from the conversation 1 time(s), 498169 characters in all (earlier reasoning, and what showed the model before its latest change). The trajectory file keeps everything.

ProVerif: 153 proofs run, 147 decided, 6 without an answer, 34 s, 57 answers recorded.

### What was analysed

| | |
|---|---|
| **Protocol** | TLS composite ML-DSA authentication |
| **Version** | draft-ietf-tls-composite-mldsa-00 |
| **Analysed** | In the fixed draft, the ML-DSA+Ed25519 composite signature and X25519MLKEM768, with full 1-RTT server authentication; every component is verified together with the correct TLS context, and the actual signature scheme follows the draft; PKI construction, HRR/PSK/0-RTT are excluded. |
| **Not analysed** | The symbolic model does not prove the computational security or post-quantum security of the algorithms; attack traces hold only under explicit assumptions and are not directly equivalent to specification vulnerabilities. Each property must have a reachability/positive-example control. |
| **Source text** | 564 numbered passages (149815 bytes) from the sections listed below. A reference such as s0001 in this report is one of these passages |

The sections of the documents the analysis is built from:

| Document | Section | Title |
|---|---|---|
| main | 2 | ML-DSA SignatureSchemes |
| main | 3 | Signature Algorithm Restrictions |
| main | 4 | Selection Criteria for Composite Signature Algorithms |
| main | 4.1 | Mapping TLS SignatureSchemes to Composite ML-DSA |
| main | 5 | Security Considerations |
| rfc9846 | 4 | Handshake Protocol |
| rfc9846 | 4.1 | The Transcript Hash |
| rfc9846 | 4.2 | Key Exchange Messages |
| rfc9846 | 4.2.1 | Cryptographic Negotiation |
| rfc9846 | 4.2.2 | Client Hello |
| rfc9846 | 4.2.3 | Server Hello |
| rfc9846 | 4.2.4 | Hello Retry Request |
| rfc9846 | 4.3 | Extensions |
| rfc9846 | 4.3.1 | Supported Versions |
| rfc9846 | 4.3.2 | Cookie |
| rfc9846 | 4.3.3 | Signature Algorithms |
| rfc9846 | 4.3.4 | Certificate Authorities |
| rfc9846 | 4.3.5 | OID Filters |

| Document | Section | Title |
|---|---|---|
| rfc9846 | 4.3.6 | Post-Handshake Certificate-Based Client Authentication |
| rfc9846 | 4.3.7 | Supported Groups |
| rfc9846 | 4.3.8 | Key Share |
| rfc9846 | 4.3.8.1 | Diffie-Hellman Parameters |
| rfc9846 | 4.3.8.2 | ECDHE Parameters |
| rfc9846 | 4.3.9 | Pre-Shared Key Exchange Modes |
| rfc9846 | 4.3.10 | Early Data Indication |
| rfc9846 | 4.3.11 | Pre-Shared Key Extension |
| rfc9846 | 4.3.11.1 | Ticket Age |
| rfc9846 | 4.3.11.2 | PSK Binder |
| rfc9846 | 4.3.11.3 | Processing Order |
| rfc9846 | 4.4 | Server Parameters |
| rfc9846 | 4.4.1 | Encrypted Extensions |
| rfc9846 | 4.4.2 | Certificate Request |
| rfc9846 | 4.5 | Authentication Messages |
| rfc9846 | 4.5.1 | Certificate |
| rfc9846 | 4.5.1.1 | OCSP Status and SCT Extensions |
| rfc9846 | 4.5.1.2 | Certificate Selection |
| rfc9846 | 4.5.1.3 | Receiving a Certificate Message |
| rfc9846 | 4.5.2 | Certificate Verify |
| rfc9846 | 4.5.3 | Finished |
| rfc9846 | 4.6 | End of Early Data |
| rfc9846 | 4.7 | Post-Handshake Messages |
| rfc9846 | 4.7.1 | New Session Ticket Message |
| rfc9846 | 4.7.2 | Post-Handshake Authentication |
| rfc9846 | 4.7.3 | Key and Initialization Vector Update |
| rfc9846 | 7 | Cryptographic Computations |
| rfc9846 | 7.1 | Key Schedule |
| rfc9846 | 7.2 | Updating Traffic Secrets |
| rfc9846 | 7.3 | Traffic Key Calculation |
| rfc9846 | 7.4 | Asymmetric Shared Secret Calculation |
| rfc9846 | 7.4.1 | Finite Field Diffie-Hellman |
| rfc9846 | 7.4.2 | Elliptic Curve Diffie-Hellman |
| rfc9846 | 7.5 | Exporters |
| rfc10024 | 4 | Negotiated Groups |
| rfc10024 | 4.1 | Client Share |
| rfc10024 | 4.2 | Server Share |
| rfc10024 | 4.3 | Shared Secret |

The purpose of the protocol as the source step located it in the documents: in the fixed draft, the ML-DSA+Ed25519 composite signature and X25519MLKEM768, with full 1-RTT server authentication; every component is verified together with the correct TLS context, and the actual signature scheme follows the draft; PKI construction, HRR/PSK/0-RTT are excluded.

## The properties against the documents

Before a query was written, the analyst set the drafted list of properties against what the documents themselves say the protocol is for. The name, what a property is about and what the documents say are the analyst's reading; the places named exist in the documents.

| ID | Name | About | What the documents say | Where |
|---|---|---|---|---|
| A02 | Live server run behind every acceptance | The client is assured that the server whose CertificateVerify it accepts holds the private key of the pinned composite certificate and ran this handshake. | The peer MUST verify the CertificateVerify signature with the negotiated composite scheme (s0019); CertificateVerify gives explicit proof that an endpoint possesses the private key of its certificate and integrity for the handshake so far (s0406); the end-entity certificate MUST carry a public key compatible with the negotiated scheme (s0020). | s0019, s0020, s0406 |
| A03 | Agreement on peers, scheme, secret, transcript | Client and server agree on the two identities, the negotiated composite SignatureScheme, the X25519MLKEM768 shared secret and the handshake transcript when the client accepts the server. | CertificateVerify is a signature over Transcript-Hash(Handshake Context, Certificate) (s0357, s0409, s0410) made with the negotiated composite scheme (s0018); for X25519MLKEM768 the shared secret is the concatenation of the ML-KEM and X25519 secrets (s0561). | s0018, s0357, s0410, s0561 |
| A04 | One acceptance per server run | Every acceptance of server authentication corresponds to a distinct server run: a CertificateVerify signed once cannot be replayed into another connection, even when the attacker can produce a different valid signature over an already signed transcript. | The handshake transcript signed in each TLS connection is unique, so impersonation requires forging a signature over the transcript of a new connection; losing SUF-CMA only lets an attacker produce a different valid signature over a transcript that has already been signed, which cannot be reused in another connection (s0046). | s0046, s0357 |
| A08 | Scheme, key, context and transcript binding | The signature scheme named in the CertificateVerify, the key type of the certificate, the TLS context string and the transcript that are checked all correspond to what the client offered and to this connection. | Both signing and verification MUST use an empty composite context string, TLS supplies domain separation by the protocol-specific context string plus transcript (s0017); the server signature context string separates signatures of different contexts against cross-protocol attacks (s0417); the algorithm MUST be one the client offered (s0423) and compatible with the certificate key (s0425). | s0017, s0417, s0423, s0425 |
| K01 | Secrecy of keys from the network | The server's composite long-term signing private key, and the handshake and application traffic secrets of an honest session, are kept from an attacker who has the network, and from an attacker who additionally recovers the X25519 half of the key agreement with a CRQC. | CertificateVerify proves possession of the private key without revealing it (s0406); the handshake combines its input secrets with the transcript into the working keying material (s0489, s0499); for X25519MLKEM768 the shared secret concatenates the ML-KEM and X25519 secrets, so one broken half does not expose it (s0561, rfc10024:299). | s0406, s0489, s0561 |
| C01 | Forward secrecy of the traffic keys | A leak of the server's long-term composite signing private key after a session completed does not expose that session's handshake and application traffic keys. | The draft argues that dropping to one secure component keeps authentication during the transition (s0045) and that composite schemes need not be discontinued once CRQCs arrive (s0047); TLS forward secrecy limits the effect of key leakage in one direction (rfc9846:6545, rfc9846:6562). | s0045, s0047, s0489 |

| ID | Name | About | What the documents say | Where |
|---|---|---|---|---|
| C08 | Authentication survives one broken component | When exactly one component private key of the server's composite certificate (traditional or ML-DSA) is broken, the client that requires composite authentication still accepts only the real server. | The traditional component contributes defense-in-depth, maintaining TLS authentication security as long as at least one component algorithm remains secure (s0045); a CRQC lets an attacker derive the traditional private key but that alone does not defeat composite authentication (s0048). | s0045, s0046, s0048 |
| R05 | No downgrade of scheme or version | The signature scheme and protocol version under which the client accepts a CertificateVerify meet the draft's policy: no RSA-PKCS1-v1_5 composite scheme in CertificateVerify, no scheme of this document in TLS 1.2, no fallback to stand-alone ML-DSA. | The pkcs15 composite schemes MUST NOT be used with "signature_algorithms" and are not defined for signed handshake messages, and a peer receiving such a CertificateVerify MUST abort (s0025, s0026); the schemes MUST NOT be used in TLS 1.2 (s0023); preventing downgrade to stand-alone ML-DSA is a matter of relying-party policy (s0052). | s0023, s0025, s0026, s0052 |
| C08.9 | One leaked component keeps authentication | The client still accepts only the real server when one component private key of the composite certificate has leaked during the session. | Authentication security is maintained as long as at least one component algorithm remains secure (s0045); a broken traditional component alone does not defeat composite authentication (s0048). | s0045, s0048 |
| A08.10 | Signer role, context and transcript bound | The CertificateVerify the client accepts was made by the server of this connection, over the server context string and over this connection's transcript, and not under another role or protocol context. | The context string for a server signature is "TLS 1.3, server CertificateVerify", used to separate signatures made in different contexts against cross-protocol attacks (s0417); the composite context string MUST be empty because TLS supplies domain separation (s0017); the signature covers Transcript-Hash(Handshake Context, Certificate) (s0410). | s0017, s0410, s0417 |
| A08.11 | Scheme compatible with certificate key | The signature scheme the client accepts is compatible with the public key in the end-entity certificate, and a mismatch is refused rather than accepted. | The end-entity certificate MUST contain a public key compatible with the negotiated SignatureScheme (s0020); the certificate key MUST be usable with a scheme in the peer's signature_algorithms (s0387) and the algorithm MUST be compatible with the certificate key (s0425). | s0020, s0387, s0425 |
| X01 | Traditional-only fallback impersonation | A client that accepts server authentication with a certificate carrying only the traditional component key, instead of the composite certificate. | A client that accepts server authentication using a traditional certificate remains vulnerable even if it also supports composite authentication: once CRQCs are available an attacker derives the private key corresponding to the traditional public key in the peer's certificate and impersonates that peer (s0048); the same concern in the other direction is a matter of relying-party policy (s0052); stripping one component works because the traditional component verifies on its own (rfc10073:2226). | main:558, main:585, main:596, s0048, s0052 |

Added from the documents by the analyst: X01.

## The security statements of the documents, paragraph by paragraph

The host listed the 72 paragraphs of the sections in which the documents say what the protocol protects and against what, and the analyst said of each what it made of it: a property, an attacker, or why neither. Taken up: 25. Not taken: 47. Which paragraphs are listed follows from the layout of the standard; where each went is the analyst's judgement.

| Paragraphs | Where | Taken up as | Why not |
|---|---|---|---|
| P01 | main:509-510 | No goal stated | A pointer to Section 11 of [I-D.ietf-lamps-pq-composite-sigs]; its substance is the composite signature considerations of RFC 10073 Section 9, mapped paragraph by paragraph at P19..P46. |
| P02 | main:520-520 | No goal stated | Page footer of the draft, no content. |
| P03 | main:524-524 | No goal stated | Running page header of the draft, no content. |

| Paragraphs | Where | Taken up as | Why not |
|---|---|---|---|
| P04 | main:527-533 | C08, C08.9, `comp_traditional`, `comp_mldsa` | |
| P05 | main:535-551 | A04, `suf_cma` | |
| P06 | main:553-556 | C01, `ltk_after_session` | |
| P07 | main:558-571 | X01, `comp_traditional` | |
| P08 | main:578-578 | No goal stated | Page footer of the draft, no content. |
| P09 | main:582-582 | No goal stated | Running page header of the draft, no content. |
| P10 | main:585-594 | X01, `comp_traditional` | |
| P11 | main:596-604 | R05 | |
| P12 | rfc10024:268-271 | A03, A08.10, `cross_protocol` | |
| P13 | rfc10024:273-276 | The prover cannot be asked | Side-channel resistance of implementations; the prover models primitives as ideal and has no notion of timing or leakage through implementation behaviour. |
| P14 | rfc10024:278-280 | No goal stated | States that the groups use fixed-length public keys, ciphertexts and shared secrets; a formatting property of the encodings. |
| P15 | rfc10024:282-287 | The prover cannot be asked | Concerns the randomness of ML-KEM encapsulation and what it carries about the generator; the symbolic model treats fresh randomness as ideal. |
| P16 | rfc10024:289-297 | The prover cannot be asked | The RNG-state attack needs a weak random number generator; RNG strength cannot be asked of the prover. |
| P17 | rfc10024:299-306 | K01, `crqc_x25519` | |
| P18 | rfc10024:308-310 | The prover cannot be asked | Shared state of one insecure RNG across the two algorithms; RNG quality, not askable in the symbolic model. |
| P19 | rfc10073:2083-2084 | No goal stated | A pointer to [RFC9881]: the ML-DSA component's own security considerations, which are computational and not askable; the composite-level consequences are mapped at P23..P39. |
| P20 | rfc10073:2088-2090 | C08, C08.9, `comp_traditional`, `comp_mldsa` | |
| P21 | rfc10073:2092-2109 | A02, C08, C08.9, `comp_traditional`, `comp_mldsa` | |
| P22 | rfc10073:2111-2126 | C08, `comp_traditional`, `comp_mldsa` | |
| P23 | rfc10073:2130-2142 | The prover cannot be asked | The prohibition on reusing component key material between composite and non-composite keys, and the X.509 trust assumptions, are key-management and PKI configuration matters outside a handshake run; the part a run can show, that a signature is bound to its context, is target A08.10. |
| P24 | rfc10073:2144-2148 | C08, C08.9, `comp_traditional`, `comp_mldsa` | |
| P25 | rfc10073:2152-2156 | No goal stated | Definition of EUF-CMA, no property of this protocol. |
| P26 | rfc10073:2158-2164 | A02, C08, `comp_traditional`, `comp_mldsa` | |

| Paragraphs | Where | Taken up as | Why not |
|---|---|---|---|
| P27 | rfc10073:2166-2167 | C08, C08.9, `comp_traditional`, `comp_mldsa` | |
| P28 | rfc10073:2169-2170 | C08, C08.9, `comp_traditional` | |
| P29 | rfc10073:2172-2173 | C08, C08.9, `comp_mldsa` | |
| P30 | rfc10073:2175-2180 | Outside the scope | About forging Composite ML-DSA signed X.509 certificates, i.e. certificate issuance and certification paths, which the task excludes; the same both-components property for the handshake signature is target C08. |
| P31 | rfc10073:2184-2189 | No goal stated | Definition of SUF-CMA, no property of this protocol. |
| P32 to P35 | rfc10073:2191-2212 | A04, `suf_cma` | |
| P36 | rfc10073:2216-2219 | No goal stated | Definition of weak non-separability, no property of this protocol. |
| P37 | rfc10073:2221-2224 | No goal stated | Definition of strong non-separability, no property of this protocol. |
| P38 | rfc10073:2226-2239 | A08.10, X01, `comp_traditional` | |
| P39 | rfc10073:2241-2248 | A08.11 | |
| P40 | rfc10073:2252-2259 | The prover cannot be asked | Best practice for generating keys; a key-management matter the prover cannot be asked about. |
| P41 | rfc10073:2261-2273 | A08.10, X01, `cross_protocol`, `comp_traditional` | |
| P42 | rfc10073:2275-2286 | Outside the scope | Revocation checking by certification authorities, part of PKI which the task excludes. |
| P43 | rfc10073:2288-2294 | A08.10 | |
| P44 | rfc10073:2298-2305 | A08.10, A08 | |
| P45 | rfc10073:2309-2313 | Outside the scope | Organisational policy for deprecating algorithms and OIDs; a management procedure, not a property of a handshake. |
| P46 | rfc10073:2315-2326 | Outside the scope | Bills of materials and module policy for deprecated component algorithms; an organisational matter outside the handshake. |
| P47 to P71 | rfc5280:5226-5449 | Outside the scope | The whole of RFC 5280 Section 8 is about certificate and CRL profiles, CA practice, revocation, name comparison and path validation; the task excludes PKI construction, and none of it states a property of the TLS handshake in this scope. |
| P72 | rfc9846:4676-4677 | No goal stated | A pointer to Appendices C, E and F of RFC 9846; the considerations in scope there (transcript and authentication, forward secrecy) are cited in the bases of A03, A08.10 and C01. |

## The catalogue of properties, entry by entry

The properties were drafted from a catalogue of security and privacy properties. The draft left out 34 of its entries, and the analyst went through each of them: examined here as a property, or left out with the reason. Examined: 9. Left out: 25. That the documents do not speak of a property is not a reason the analyst may give.

| Entry | Property of the catalogue | Examined as | Why not |
|---|---|---|---|
| K02 | Payload and end-to-end confidentiality | K01 | |
| K03 | Strong secrecy / indistinguishability of secret values | The prover cannot be asked | Strong secrecy is asked as the attacker telling two allowed values apart; here the application traffic secret is derived from fresh ephemeral randomness rather than chosen among two candidates, so there is no pair of values to distinguish and noninterf has no instantiation. K01 asks the closest askable property. |
| K04 | Key purpose, direction and session separation | Outside the scope | Key purpose, direction and epoch separation belong to the TLS 1.3 key schedule (s0488 to s0513), which the task keeps out of scope: the model derives one handshake and one application secret from the X25519MLKEM768 shared secret, not the direction and epoch hierarchy. |
| D01 | Message integrity and authenticity of data origin | A03, A08 | |
| A01 | Peer participation (Aliveness) | A02 | |
| A05 | Implicit key authentication | A03, K01 | |
| A06 | Explicit key confirmation | A03 | |
| A07 | Identity–credential–key binding | Outside the scope | Binding an identity to a credential and a key is certificate-chain and PKI work, which the task excludes; the model pins the server's composite public key and does not build or validate a chain. |
| R01 | Authentication / challenge freshness | A04 | |
| R02 | Message-level replay resistance | A04 | |
| R03 | At most one business operation | Not applicable | The protocol carries no billing, control or consumable token operation. |
| R04 | Message acceptance order / window safety | Not applicable | The analysed flow carries no acceptance window, sequence number or counter: the handshake messages are exchanged once in a fixed order. |
| C02 | Resistance to key-compromise impersonation (KCI) | Not applicable | Only the server is authenticated in this flow; the client holds no long-term credential in scope, so there is no own long-term key whose leak could enable impersonating the peer. |
| C03 | Known-key security / compromise isolation | C01 | |
| C04 | Resilience to ephemeral secret / partial state compromise | C08, K01 | |
| C05 | Post-compromise security (PCS) | Outside the scope | The analysed run is a single full 1-RTT handshake with no resumption, rekeying or ratchet; the task excludes PSK resumption and 0-RTT, so there is no later session whose confidentiality could be regained. |
| C06 | Enhanced authentication after compromise of the registration credential | Not applicable | There is no password-authenticated or augmented registration record in the protocol. |
| C07 | Resistance to offline dictionary attacks | Not applicable | The protocol uses no password or other low-entropy secret, so no offline dictionary attack is possible. |
| Z01 | Correct enforcement of authorization | Not applicable | TLS 1.3 authentication establishes identities and keys; it carries no authorization decision or protected operation. |
| Z02 | Delegated authority does not expand | Not applicable | No delegation or proxy mechanism is used. |
| Z03 | Authorization bound to identity / session / operation | Not applicable | No authorization context is conveyed or bound. |
| Z04 | Revocation and expiry take effect | Not applicable | The protocol carries no permit with an expiry or revocation. |
| Z05 | Cross-tenant / cross-service security isolation | Not applicable | No tenant or multi-service isolation is part of the handshake. |

| Entry | Property of the catalogue | Examined as | Why not |
|---|---|---|---|
| S01 | Encryption nonce uniqueness | Outside the scope | The AEAD nonce construction is the TLS 1.3 record layer's (s0519 to s0529), not part of the composite ML-DSA authentication flow the task analyses. |
| S02 | Retransmission / key reinstallation safety | Not applicable | TLS 1.3 has no retransmission and no key reinstallation message; the handshake keys are installed once per handshake. |
| S03 | Resistance to rollback and state monotonicity | Not applicable | The handshake keeps no cached version, counter or resumable state in this scope. |
| S04 | Cross-phase security context consistency | Outside the scope | Cross-phase context consistency concerns resumption, HelloRetryRequest and 0-RTT, all excluded by the task. |
| S05 | Termination and invalidation of old state | Outside the scope | Teardown and closure are not part of the authentication flow the task analyses. |
| P01 | Identity anonymity | Not applicable | Server authentication transmits the server's certificate in the clear and the client sends no identity at all; the protocol makes no attempt to hide which party takes part. |
| P02 | Session unlinkability | Not applicable | The server always presents the same certificate, so sessions are linkable to that server by design. |
| P03 | Hiding of user / credential existence | Not applicable | No user name or credential is looked up in this flow, so there is no existence to hide. |
| V01 | Conditional progress / error recovery | The prover cannot be asked | Whether a run can always complete or recover from an error is a progress property; the prover cannot be asked that a run always ends. |
| V02 | No unexpected deadlock | The prover cannot be asked | Absence of deadlock is a branching-time progress property, which the prover cannot be asked. |
| V03 | Bounded resource abuse / amplification resistance | The prover cannot be asked | Resource consumption beyond counting events cannot be asked of the prover. |

## The parties and the protected values, one by one

The host listed the 15 parties and protected values that the drafted properties name, and the analyst said of each which attacker controls the party or holds the value, or why no such attacker was set. With an attacker: 3. Without: 12. The list follows from the properties; what was made of each entry is the analyst's judgement.

| ID | Party or value | Kind | Attacker | Why none |
|---|---|---|---|---|
| S01 | Client | party | Nothing gained | The client holds only the server's public composite certificate and its own ephemeral secrets; the properties here are about what the server assures the client, so a dishonest client gains nothing the network attacker lacks. |
| S02 | Server | party | Everything lost by definition | A dishonest server is the very party the client is authenticating: every authentication property holds trivially for it and every secrecy property is lost by definition, and the documents ask no property under a malicious server. |
| S03 | Server identity | value | Nothing gained | The server's identity is public: it is the name in the certificate and in the certificate itself. |
| S04 | Negotiated signature scheme | value | Nothing gained | The negotiated signature scheme is a public constant of the connection. |
| S05 | Handshake transcript | value | Nothing gained | The handshake transcript is on the wire in the clear; the attacker already has it. |
| S06 | Key exchange shared secret | value | Everything lost by definition | The traffic-key properties are lost by definition once the shared secret is known, and no document asks a key-agreement property under a full break of it; the only documented partial break, a CRQC recovering the X25519 scalar (rfc10024:299), is the attacker crqc_x25519, set for K01 and A02. |

| ID | Party or value | Kind | Attacker | Why none |
|---|---|---|---|---|
| S07 | Signature scheme | value | Nothing gained | The signature scheme identifier is a public constant. |
| S08 | Certificate public key | value | Nothing gained | The certificate public key is public; only its private half matters and that is value S10. |
| S09 | context string | value | Nothing gained | The context string is a public constant of the protocol. |
| S10 | Server long-term signing private key | value | `ltk_after_session` | |
| S11 | Handshake traffic key | value | `session_state` | |
| S12 | Application traffic key | value | `session_state` | |
| S13 | Composite signature | value | Nothing gained | The composite signature travels on the wire in the CertificateVerify message. |
| S14 | Protocol version | value | Nothing gained | The protocol version is a public field of the hello messages. |
| S15 | Signer role | value | Nothing gained | The signer role is fixed by the context string, a public constant. |

## The properties under each attacker

The attackers are the analyst's, set before a query was written; each answer was recorded under one of them. A cell is the prover's answer for the property under that attacker; "No answer" is an attacker set for the property under which the prover left a claim undecided or nothing was recorded; an empty cell is an attacker the analyst did not set for the property.

| Attacker | Has | Basis | Where |
|---|---|---|---|
| `network` | the network only: it reads, drops, changes and sends messages, and holds no secret of a party | | |
| `comp_traditional` | the private key of the traditional component (Ed25519/ECDSA) of the server's composite certificate, obtained once CRQCs are available, and can sign with that component only | Once CRQCs are available, an attacker can derive the private key corresponding to the traditional public key in the peer's certificate and impersonate that peer (s0048); the traditional component is only defense-in-depth (s0045). | s0045, s0048 |
| `comp_mldsa` | the private key of the ML-DSA component of the server's composite certificate, and can sign with that component only | Composite authentication is used precisely because ML-DSA is not fully trusted on its own, so breaking the ML-DSA component must not break composite authentication (s0052, s0045). | s0045, s0052 |
| `ltk_after_session` | the server's complete composite long-term signing private key (both components), obtained after the analysed session has completed | The draft plans migration away from composite ML-DSA once CRQCs arrive (s0045, s0047) and TLS forward secrecy limits the effect of key leakage at a later time (rfc9846:6545, rfc9846:6562). | s0045, s0047 |
| `suf_cma` | the ability to produce a different valid signature over a message which the honest server has already signed, i.e. the composite signature is not strongly unforgeable | A composite signature is SUF-CMA only if every component is; ECDSA is not, and once CRQCs are available no composite scheme except those with RSA-PKCS1-v1_5 is (s0046); an ECDSA signature can be trivially modified into another valid signature for the same message, and this passes through to the composite (rfc10073:2191). | s0046 |
| `cross_protocol` | composite ML-DSA signatures made with the same long-term key in another protocol or role context, obtained by key reuse or cross-protocol interaction | Key reuse between composite and single-algorithm contexts is forbidden because it leads to stripping attacks and further cross-protocol risks (rfc10073:2261, rfc10073:2272); TLS separates signature contexts by the protocol context string plus transcript (s0017, s0417). | s0017, s0417 |

| Attacker | Has | Basis | Where |
|---|---|---|---|
| `crqc_x25519` | the X25519 ephemeral scalars of a completed session, hence the X25519 half of the X25519MLKEM768 shared secret, obtained with a CRQC | Any passive observer with access to a cryptographically relevant quantum computer can recover the ECDH ephemeral scalar, which is derived directly from RNG output (rfc10024:299). | s0561 |
| `session_state` | the handshake and application traffic secrets of the session, read from the memory of one of the parties during the session | I consider it because the exposure of one party has to be asked: a party whose session state is read holds the handshake and application traffic secrets (s0489, s0519), and what remains is what the other side is assured of. | s0489, s0519 |

| ID | `network` | `comp_traditional` | `comp_mldsa` | `ltk_after_session` | `suf_cma` | `cross_protocol` | `crqc_x25519` | `session_s` |
|---|---|---|---|---|---|---|---|---|
| A02 | Holds | Holds | Holds | | Holds | | Holds | Holds |
| A03 | Holds | | | | | | | |
| A04 | Holds | | | | Holds | | | |
| A08 | Holds | | | | | Holds | | |
| K01 | Holds | | | | | | Holds | |
| C01 | | | | Holds | | | | |
| C08 | | Holds | Holds | | | | | Holds |
| R05 | Holds | | | | | | | |
| C08.9 | | Holds | Holds | | | | | |
| A08.10 | Holds | | | | | Holds | | |
| A08.11 | Holds | | | | | | | |
| X01 | | Fails | | | | | | |

## Source and targets

Source: 564 units, 149815 bytes, frozen in source-delivery-01.json. Targets: 8 approved (A02, A03, A04, A08, K01, C01, C08, R05); the questions asked: 3. The list of targets was approved as drafted, beforehand, with --approve-targets.

Decisions the host took by its documented default (task `decisions: auto`), for you to check:

- approve_package: approved source package protocol-b387105272c6f7bd98b9d23cc2dc

This step as it reports itself: 1 requests (1 failed), 0.000 USD, 69677 tokens in and 4040 out, 27 s.

## Notes on the prover and the model

What the analyst named for targets:

| step | targets | features | reason |
|---|---|---|---|
| 52 | A02, A03, A04, A08, A08.10, A08.11, K01, C01, C08, C08.9, R05, X01 | plain_primitives, many_sessions | The model uses only signatures (two component algorithms, composed), a KEM, Diffie-Hellman as a plain key agreement with the single law g^xy = g^yx, hashing, an AEAD and a keyed derivation; there is no exclusive-or and no counter or replaced key that a property depends on, so no bounded.pv is written. Two idealisations the answers rest on: the key schedule and the Finished MAC are private (one-way) functions, so the attacker cannot compute the traffic keys even when it knows the shared secret, and the handshake flight is sent without its record-layer encryption. All targets are asked over any number of client and server sessions. |

## The analyst's closing note

All twelve targets are answered on one ProVerif model (model.pv) with controls: A02, A03, A04, A08, A08.10, A08.11, K01, C01, C08, C08.9 and R05 are proved; X01 (the traditional-only fallback the draft itself warns about, s0048) is the one attack, marked recorded at main:558 and drawn as a figure. The result worth quoting: with exactly one component private key broken, with the X25519 half recovered by a CRQC, with the session keys read and with a same-key signature from another protocol context, the client accepts a server CertificateVerify only when the real server signed this connection's transcript with the negotiated scheme and the pinned composite key type, and no RSA-PKCS1-v1_5 composite scheme, stand-alone ML-DSA or TLS 1.2 connection is accepted. The recommendation on X01 was checked in a copy of the model: with the second client requiring the composite certificate and both component signatures, the impersonation is gone, the normal run survives and all other answers are unchanged. What limits this: the handshake flight is modelled without its record-layer encryption and the key schedule and Finished MAC are one-way, so the proofs are about the signature and the transcript, not the record layer or a shared-secret leak, and the broken-component attacker gets one component signature over one message rather than a signing oracle. No target rests on a counter or a replaced key, so no bounded.pv and no state review were due. Next step: a Tamarin+ Agent model with the record layer and public KDFs, to see whether the same holds when the attacker can play a party end to end; attempts to make those functions public here made ProVerif time out.