Security analysis of TLS 1.3 pure ML-KEM-768 1-RTT

This report in another language: 中文.

draft-ietf-tls-mlkem-11 · ProVerif+ Agent report

Documents	ML-KEM Post-Quantum Key Agreement for TLS 1.3 (protocol definition); The Transport Layer Security (TLS) Protocol Version 1.3 (supporting)
Properties examined	15: 12 hold, 3 fail (3 attacks)
Findings	None

Contents

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

1 Protocol and scope

This is TLS 1.3 with ML-KEM-768 as the only key-establishment group, server certificate authentication and a full one round-trip handshake; HelloRetryRequest, PSK, 0-RTT and client certificates are out of scope. The client generates an ML-KEM key pair and sends the encapsulation key as its key_share; the server encapsulates to it, returns the ciphertext, derives the shared secret, and authenticates with a certificate, a CertificateVerify over the transcript, and a Finished. The client checks all three, sends its own Finished, and both sides exchange application data under keys derived from the shared secret.

When the exchange ends the client knows it is talking to the server its certificate names, because the signature covers a transcript that includes the client's own random and its own encapsulation key. Both sides know they hold the same session keys and that an attacker on the network cannot derive them. The server learns nothing about who the client is: in this scope it is assured only that the peer knows the shared secret of this handshake.

The parties:

- **Client**: wants a channel to srv. Holds: ML-KEM-768 offered; server name srv; CA public key.
- **Server**: single operator, ML-KEM-768 only. Holds: certificate (srv, pk_srv); signature private key sk_srv.

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. The table in section 3 has the result for every property.

Within the scope analysed the protocol gives what it is meant to give: server authentication, agreement on the key shares and on the session keys, secrecy of those keys against the network, and forward secrecy with respect to the server's signature key all hold for any number of sessions. No finding: the three failing results reproduce weaknesses the documents state themselves, namely that the ML-KEM decapsulation key and the encapsulation randomness must be erased because whoever holds them derives the session, and that a server which authenticates only itself cannot distinguish a legitimate client from anyone else. The results rest on a symbolic ML-KEM, on a public certificate directory and on a transcript modelled as a hash chain; the record layer, the encoding checks and the computational security of ML-KEM are outside them. Tamarin plus Agent, which carries real algebraic and state models, would be needed to settle the parts this abstraction removes.

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	**server authentication to the client.** When the client accepts handshake completion, there exists a server run whose intended communication partner is the client's identity.	no compromise	Holds	medium	no
A03	**matching server run agreement.** When the client accepts, there exists a matching server run, and the two identities, the negotiated ML-KEM-768 group, the key shares and the shared secret agree.	no compromise	Holds	medium	no
A04	**injective agreement on the server run.** Every time the client accepts handshake completion, it corresponds to a distinct server run (injective matching).	no compromise	Holds	medium	no
A06	**server Finished key confirmation.** When the client verifies the server Finished, the server has derived and holds the same handshake keys for this session.	no compromise	Holds	medium	no
A08	**negotiated group and share binding.** The ML-KEM-768 group, the server key share (ciphertext) and the negotiated parameters that the client accepts correspond to what it proposed for this session and are bound by the transcript.	no compromise	Holds	medium	no
K01	**session key and shared secret secrecy.** The attacker cannot derive the ML-KEM shared secret or the derived handshake and application traffic keys.	no compromise	Holds	high	no

ID	Property	Compromise assumed	Result	Value	In the documents
C01	**forward secrecy from the server signature key.** After the session completes, leaking the server's long-term signature private key still does not let the attacker derive the session keys of previously completed sessions.	server long-term signature private key (leaked after the session completes)	Holds	high	no
C04	**session secrecy under KEM key disclosure.** After the client's ML-KEM decapsulation private key is leaked, the attacker still cannot derive the shared secret and traffic keys of that session.	client ML-KEM decapsulation private key sk (leaked during or after the session)	Attack holds (stated in the documents)	low	yes
C02	**server authentication under key compromise.** After the server's long-term signature private key is leaked, the attacker still cannot impersonate the client or a peer whose key has not leaked in order to make the server accept.	server long-term signature private key (leaked before the handshake)	Attack holds (stated in the documents)	low	yes
K01.10	**session key secrecy and agreement.** The attacker cannot derive the session keys, and both sides accept the same session keys.	no compromise	Holds	high	no
A03.11	**server identity and transcript binding.** When the client accepts, the server identity is consistent with the full handshake transcript.	no compromise	Holds	medium	no
C01.12	**forward secrecy after signature key leak.** After the session completes, leaking the server's long-term signature private key still keeps historical session keys secret; KEM private key leakage is analysed separately.	server long-term signature private key (leaked after the session completes)	Holds	high	no
X01	**application data confidentiality.** Application data carried after the handshake is not readable by an attacker who controls the network and holds nothing else.	no compromise	Holds	medium	no
X02	**session secrecy under encapsulation randomness.** Even when the attacker is given the ML-KEM encapsulation randomness drawn for a handshake, that handshake's shared secret and traffic keys must stay secret.	the encapsulation randomness that ML-KEM drew for one handshake, i.e. raw output of the RNG as described in main:295, obtained after the handshake	Attack holds (stated in the documents)	low	yes
X03	**independence of handshake and application keys.** The traffic keys derived for the handshake must not reveal the application traffic keys of the same handshake or those of another handshake.	the handshake traffic keys of one completed handshake (client_handshake_traffic_secret and server_handshake_traffic_secret, or the keys derived from them), obtained after the handshake	Holds	medium	no

"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, X02, X03.

The questions asked, and the analyst's answers:

Question	Answer	Rests on
Q1: Session key secrecy and key agreement between the two sides	Session keys stay secret against the attacker on the network and both sides end with the same ones, for any number of sessions; the only disclosures that reach a session key are the ephemeral ML-KEM secrets of that very handshake, which the documents require to be erased.	K01, K01.10, C04, X02
Q2: Server authentication and full transcript binding of the Finished	The server is authenticated because CertificateVerify signs a transcript that includes the client's random and its encapsulation key, and the client accepts only a Finished under keys derived from that transcript and from the ML-KEM shared secret.	A02, A03, A03.11, A08, A06
Q3: Separate analysis of a later leak of the long-term signature key and of KEM key leakage	A later theft of the server's signature key does not open past or current sessions, because the session keys come from the ML-KEM secret alone; the ML-KEM decapsulation key and the encapsulation randomness are a separate exposure and do open the session, as the documents warn where they require those values to be erased.	C01, C01.12, C04, X02

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:

- ML-KEM is modelled as an abstract encapsulation/decapsulation pair: its computational security, its implicit rejection, and the malformed-key and ciphertext-length checks of main s0026 to s0028 are not modelled.
- Only ML-KEM-768 is modelled; cipher-suite, version and multiple-group negotiation, HelloRetryRequest, PSK and 0-RTT are out of scope.
- The record layer is not modelled: no sequence numbers, alerts, padding, key update or length hiding, so nothing is said about traffic analysis or record-level replay.
- The transcript is a hash chain of message contents, not the byte-exact Transcript-Hash with handshake headers and record framing.
- The certificate is a public value from a directory, so the confidentiality of the Certificate message, and F.1's protection of endpoint identities, are not asked.
- Post-compromise security, that is a handshake ordered after the long-term key leak, is not asked: the phases put every handshake before the disclosure.
- No target rests on a counter, sequence number or replaceable key, so no value that changes across steps was modelled and no bounded model or state review was done.
- Availability, timing, side channels and the cost of a request are outside what the traces of this model can say.

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: server authentication to the client

When the client accepts that the handshake is complete, there exists a server run whose intended peer is the client identity.

Every time a client accepts, a server run of that very handshake exists that sent the Finished for the same ML-KEM ciphertext, and the certificate the client accepted names the server it asked for. The claim's words about the server's intended peer have no counterpart in the protocol: TLS 1.3 authenticates the server only, so the client's assurance is about the server's identity.

- Result: Holds; answers what is asked: partly; usable with caveats
- Value medium: Peer authentication is the central promise of the protocol and it holds for any number of sessions; the value is reduced because the certificate and the transcript are idealised. I searched the documents for a statement that server authentication fails and found none: F.1 states it as a goal.
- Stated in the documents: no
- Rests on: The certificate comes from a public directory and the CA key is trusted by assumption; path validation and name matching are outside the model.; Only ML-KEM-768 is modelled, so nothing is said about negotiation among groups.

A03: matching server run agreement

When the client accepts, a matching server run exists, and both sides agree on the identities, the negotiated ML-KEM-768 group, the key shares and the shared secret.

A client that accepts shares the group identifier, its own encapsulation key, the ciphertext and the ML-KEM shared secret with a server run of the same handshake. No network attacker can make the two sides end with different shares or secrets.

- Result: Holds; answers what is asked: yes; usable
- Value medium: Agreement on the key-schedule inputs is what the ML-KEM substitution has to preserve, and it holds for any number of sessions.
- Stated in the documents: no
- Rests on: The transcript is an explicit hash chain of message contents rather than the byte-exact TLS transcript hash.

A04: injective agreement on the server run

Every client acceptance of a completed handshake corresponds to a distinct server run (injective matching).

A recorded server flight cannot be presented to a second client acceptance: the ciphertext is encapsulated to the encapsulation key that client sent, so every acceptance belongs to its own server run and per-handshake keys are fresh.

- Result: Holds; answers what is asked: yes; usable

- Value medium: Injective agreement is what makes the freshness of the per-handshake keys usable, and it holds without a simplification that touches it.
- Stated in the documents: no
- Rests on: No resumption is modelled, so the handshake flight is the only replay surface.

A06: server Finished key confirmation

When the client verifies the server Finished, the server has derived and holds the same handshake keys for this session.

When the client accepts the server Finished it already holds the handshake traffic keys, and the server of that run derived the same ones, so the Finished is genuine key confirmation.

- Result: Holds; answers what is asked: yes; usable
- Value medium: Key confirmation is a stated goal of the Finished message and it holds for any number of sessions.
- Stated in the documents: no
- Rests on: The Finished is a MAC under a finished_key derived from the handshake traffic secret, not a real HMAC.

A08: negotiated group and share binding

The ML-KEM-768 group, the server key share (ciphertext) and the negotiated parameters that the client accepts correspond to what it proposed in this session and are bound by the transcript.

The group identifier, the client's encapsulation key and the server's ciphertext that the client accepts all belong to a server run of that same handshake, so the key schedule commits to the values actually exchanged and the share cannot be swapped.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value medium: This is what makes re-encapsulation attacks inapplicable and it is shown mechanically; the caveat is the two omitted encoding checks.
- Stated in the documents: no
- Rests on: The FIPS 203 encapsulation-key check and the ciphertext length check are not modelled; only the group identifier and the binding of the values are.

A03.11: server identity and transcript binding

When the client accepts, the server identity is consistent with the complete handshake transcript.

The client verifies CertificateVerify with the signing key of a server run of this handshake, and that signature covers a transcript containing the client random, the client's encapsulation key, the server random and the ciphertext, so server identity and handshake transcript are tied together where the client decides.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value medium: The transcript binding of CertificateVerify carries the whole server authentication, but the transcript is idealised.
- Stated in the documents: no
- Rests on: The transcript is a hash chain of contents, not the byte-exact transcript hash including handshake headers.

K01: session key and shared secret secrecy

The attacker cannot derive the ML-KEM shared secret or the derived handshake and application traffic keys.

An attacker who controls the network and holds nothing else never learns the ML-KEM shared secret, the handshake traffic keys or the application traffic keys of a handshake the client authenticated, for any number of sessions. The falsified control shows the same query fails once one handshake secret is given away.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value high: Secrecy of the session keys is the primary goal of the key exchange and it holds for any number of sessions against the network attacker.
- Stated in the documents: no
- Rests on: ML-KEM is an abstract encapsulation/decapsulation pair without implicit rejection or algebraic structure; nothing is proved about its computational security.; Only ML-KEM-768 is modelled.

K01.10: session key secrecy and agreement

The attacker cannot derive the session keys, and both sides accept the same session keys.

Both sides of a handshake the client authenticated end with the same application traffic secrets and no network attacker can read the application data; secrecy and agreement come out together.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value high: Secret and agreed session keys is what the channel sells, and both halves hold for any number of sessions.
- Stated in the documents: no
- Rests on: The record layer is not modelled: one AEAD message per direction, no sequence numbers, no key update.

C01: forward secrecy from the server signature key

After the session completes, the server's long-term signature private key is leaked, and the attacker still cannot derive the session keys of previously completed sessions.

When the server's long-term signature key is disclosed after the handshake, the application traffic keys of the completed handshake stay secret. The model also gives the attacker that handshake's handshake traffic keys at the same step, so the result is stronger than the target asks.

- Result: Holds; answers what is asked: yes; usable
- Value high: Forward secrecy with respect to the signature key is what lets a certificate-based deployment survive a later key theft, and it holds for any number of sessions.
- Stated in the documents: no
- Rests on: The compromise happens after every handshake of the run, which is the timing the target names.; The extra disclosure of the handshake traffic keys is my choice, not part of the target.

C04: session secrecy under KEM key disclosure

After the client's ML-KEM decapsulation private key is leaked, the attacker still cannot derive the shared secret and traffic keys of that session.

Once the client's ML-KEM decapsulation key of a handshake is disclosed, the attacker recovers the encapsulation randomness from the ciphertext and derives the whole key schedule of that handshake. Nothing in the protocol can prevent this: the decapsulation key is the only thing standing between the ciphertext and the shared secret, which is why the documents require it to be erased.

- Result: Attack holds (stated in the documents); answers what is asked: yes; usable
- Value low: It follows directly from the exposure the target assumes; the documents state the same consequence when they require ephemeral keys to be erased.
- Stated in the documents: yes. Where the documents say it: rfc9846:6511
- Rests on: The decapsulation key is disclosed after the handshake, which is the timing the target names.; ML-KEM is abstract: the model shows the mechanism, not the computational hardness of the real primitive.

C02: server authentication under key compromise

After the server's long-term signing private key leaks, an attacker still cannot impersonate a client or an uncompromised peer and make the server accept it.

The trace shows the attacker sending a ClientHello with an encapsulation key whose decapsulation key it holds; the server encapsulates to it and the attacker completes both Finished messages without any leaked long-term key. That is the attacker acting as a client, which TLS 1.3 in this scope permits because no client certificate is in play. What survives the signature-key leak is key possession: the server accepts only a peer that knew that handshake's shared secret.

- Result: Attack holds (stated in the documents); answers what is asked: partly; usable
- Value low: The failure follows from the scope decision to exclude client certificates, and F.1 states that in a unilaterally authenticated connection the attacker can establish its own session keys with the server.
- Stated in the documents: yes. Where the documents say it: rfc9846:6485, rfc9846:6482
- Rests on: No client authentication exists in this scope, so the first half of the claim cannot hold for a protocol that authenticates only the server.; The record layer is not modelled, so the trace is shown up to the server's acceptance of the Finished.

C01.12: forward secrecy after signature key leak

If the server's long-term signing private key leaks after a session completes, the historical session keys stay secret; disclosure of the KEM private key is analysed separately.

Historical session keys of completed handshakes stay secret when the server's long-term signature key leaks afterwards; only the disclosure of the ephemeral ML-KEM secret of a handshake reaches that handshake's keys.

- Result: Holds; answers what is asked: yes; usable
- Value high: It answers the draft's question: a signature-key theft does not open past traffic, and the ephemeral secrets are what must be erased.
- Stated in the documents: no
- Rests on: The two exposures are separated by the phase in which the attacker holds them, in one model, not by separate processes.

X01: application data confidentiality

Application data carried after the handshake is not readable by an attacker who controls the network and holds nothing else.

Application data carried under the application traffic keys is not readable by an attacker on the network; the control shows it becomes readable once the ephemeral ML-KEM secret of that handshake is disclosed.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value medium: It is the confidentiality half of the channel, proved for any number of sessions with the record layer idealised.
- Stated in the documents: no
- Rests on: The record layer, its padding and its length hiding are not modelled, so nothing is said about what traffic analysis learns.; The payload is one fresh value per direction.

X02: session secrecy under encapsulation randomness

Even when the attacker is given the ML-KEM encapsulation randomness drawn for a handshake, that handshake's shared secret and traffic keys must stay secret.

Once the encapsulation randomness of a handshake is disclosed the attacker derives the shared secret and the traffic keys of that handshake. In ML-KEM the randomness is what the shared secret is made of, so nothing in the protocol prevents it; what prevents it is RNG hygiene and erasing the value.

- Result: Attack holds (stated in the documents); answers what is asked: yes; usable
- Value low: It follows directly from the exposure the target assumes, and the documents warn of it themselves; it is reproduced to show the model sees it.
- Stated in the documents: yes. Where the documents say it: main:295
- Rests on: The shared secret is modelled as a one-way function of the encapsulation randomness, which is the faithful reading of the statement that the decapsulator recovers that randomness exactly.; The disclosure is modelled after the handshake.

X03: independence of handshake and application keys

The traffic keys derived for the handshake must not reveal the application traffic keys of the same handshake or those of another handshake.

The handshake traffic keys of a completed handshake do not yield the application traffic keys of the same handshake, and different handshake secrets stay independent. The key-schedule labels are what separates them.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value medium: It is the reason a handshake-key leak is survivable, but the separation is a property of the KDF that the model takes as given.
- Stated in the documents: no
- Rests on: The separation rests on the symbolic treatment of HKDF labels as distinct constants; collisions in the KDF are outside what this model decides.

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	checked		1 proved
A03	proved		a03	checked		1 proved
A04	proved		a04	checked		1 proved
A06	proved		a06	checked		1 proved
A08	proved		a08	checked		1 proved
K01	proved		k01	checked		1 proved
C01	proved		c01	checked		1 proved
C04	none				c04 (holds_in_source)	1 attack
C02	none				c02 (holds_in_source)	1 attack
K01.10	proved		k10	checked		2 proved
A03.11	proved		a11	checked		1 proved
C01.12	proved		c01	checked		1 proved
X01	proved		x01	checked		1 proved
X02	none				c04 (holds_in_source)	1 attack
X03	proved		c01	checked		1 proved

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	The client accepts only if a server run of this very handshake exists: every acceptance is matched by a server that sent the Finished for the same ML-KEM ciphertext. TLS 1.3 has no client identity for the server's intended peer, so what the client is assured of is that the peer identity it checks is the server it named and that a matching server run exists.	`network`	proved	a02	checked
A03	When the client accepts, a server run of the same handshake exists with the same ML-KEM-768 group, the same encapsulation key, the same ciphertext and the same shared secret.	`network`	proved	a03	checked
A04	Every client acceptance corresponds to a distinct server run: a recorded server flight cannot be presented to a second client acceptance, because the ciphertext is bound to the client's own fresh encapsulation key.	`network`	proved	a04	checked
A06	When the client accepts the server Finished it holds the handshake traffic keys, and the server of that run has derived the same handshake traffic keys (key confirmation).	`network`	proved	a06	checked

target	claim	attacker	answer	query	control
A08	The group identifier, the client's encapsulation key and the server's ciphertext that the client accepts are those of a server run of this very handshake, so the key share the key schedule commits to cannot be swapped for another one.	`network`	proved	a08	checked
K01	The ML-KEM shared secret and the handshake and application traffic keys of a handshake that the client authenticated are not derivable by the attacker who holds only the network, for any number of sessions.	`network`	proved	k01	checked
C01	After the server's long-term signature key is disclosed (and the handshake traffic keys of the completed handshake with it), the application traffic keys of that handshake remain unknown to the attacker.	`server_sig_key`	proved	c01	checked
C04	After the client's ML-KEM decapsulation key of a handshake is disclosed, the shared secret and the traffic keys of that handshake stay secret.	`client_kem_key`	attack (holds_in_source)	c04 (the control of c01)	
C02	With the server's long-term signature key, the attacker cannot impersonate a client: every handshake the server accepts is matched by a client run that sent the Finished. What the protocol has instead of client authentication: the server is assured only of key possession.	`server_sig_key`	attack (holds_in_source)	c02 (the control of k10)	
K01.10	Both sides of a handshake that the client authenticated end with the same session keys: the client's application traffic secrets equal the server's, for the same ciphertext.	`network`	proved	k10	checked
K01.10	Secrecy of the session keys of an authenticated handshake against the attacker on the network alone, and agreement on those keys between the two sides.	`network`	proved	k01	checked
A03.11	The server identity in the Certificate and the transcript that CertificateVerify covers: the signing key the client verified is the key of a server run of this handshake, and the signature covers the client's own random, the client's encapsulation key, the server Random, the ciphertext and the certificate.	`network`	proved	a11	checked
C01.12	Historical session keys of completed handshakes stay secret when the server's long-term signature key is disclosed afterwards; the ML-KEM private key and encapsulation randomness are analysed as a separate exposure and are not disclosed here.	`server_sig_key`	proved	c01	checked
X01	Application data sent after the handshake is not readable by the attacker who holds only the network.	`network`	proved	x01	checked
X02	When the ML-KEM encapsulation randomness drawn for a handshake is disclosed, that handshake's shared secret and traffic keys stay secret.	`encap_randomness`	attack (holds_in_source)	c04 (the control of c01)	
X03	The handshake traffic keys of a completed handshake do not reveal the application traffic keys of the same handshake, nor those of another handshake.	`handshake_key`	proved	c01	checked

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-20261114T210655, 2026-11-14. Status: finished. Model: deepseek-v4-1-flash-261017. Steps: 37, prover 164 s, wall 14 min, cost 0.117 USD.

ProVerif: 43 proofs run, 43 decided, 0 without an answer, 157 s, 16 answers recorded.

What was analysed

Protocol	TLS 1.3 pure ML-KEM-768 1-RTT
Version	draft-ietf-tls-mlkem-11
Analysed	Fixed ML-KEM-768, server signature authentication and full 1-RTT; the TLS baseline is fixed at RFC9846, with the updates relative to the draft's cited version noted; includes the binding of CertificateVerify/Finished to the actual handshake transcript and encrypted handshake messages. Excludes HRR, PSK, 0-RTT, client certificates.
Not analysed	The symbolic model does not prove the computational security or post-quantum security of the algorithms; attack traces hold only under the stated assumptions and are not directly equivalent to specification vulnerabilities. Every property must have a reachability/positive-example control.
Source text	592 numbered passages (151676 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	3	Key encapsulation mechanisms
main	4	Construction
main	4.1	Negotiation
main	4.2	Transmitting encapsulation keys and ciphertexts
main	4.3	Shared secret calculation
rfc9846	2	Protocol Overview
rfc9846	2.1	Incorrect DHE Share
rfc9846	2.2	Resumption and Pre-Shared Key (PSK)
rfc9846	2.3	0-RTT Data
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

Document	Section	Title
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
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

Document	Section	Title
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

The purpose of the protocol as the source step located it in the documents: fixed ML-KEM-768, server signature authentication and full 1-RTT; the TLS baseline is fixed at RFC9846, with the updates relative to the draft's cited version noted; includes the binding of CertificateVerify/Finished to the actual handshake transcript and encrypted handshake messages. Excludes HRR, PSK, 0-RTT, client certificates.

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	server authentication to the client	the client, which must end the handshake with the peer identity being the server it asked for (rfc9846 F.1 Peer Authentication); TLS 1.3 authenticates the server only, the client side is unauthenticated in this scope	F.1 states 'Peer Authentication: The client's view of the peer identity should reflect the server's identity', against an active network attacker.	rfc9846:6490
A03	matching server run agreement	the client and the server: both must end with the same session keys, the same ML-KEM-768 group, the same encapsulation key and ciphertext as key_share, and the same shared secret	F.1 'Establishing the same session keys' and 'Downgrade Protection'; main s0023 s0024 s0025 put pk and ct into key_share and s0034 puts the ML-KEM shared secret into the key schedule.	rfc9846:6478, rfc9846:6499, s0023, s0024, s0025, s0034
A04	injective agreement on the server run	the server run behind every client acceptance: each acceptance must correspond to a distinct server handshake, so that a recorded flight cannot be presented twice	F.1 'Uniqueness of the session keys: Any two distinct handshakes should produce distinct, unrelated session keys'; the ClientHello.random and the server Random make each handshake distinct (s0535).	rfc9846:6495, rfc9846:6535, s0535
A06	server Finished key confirmation	the server, which must already hold the handshake traffic keys of this session before the client accepts its Finished	F.1 'Establishing the same session keys'; s0063 the Finished is a MAC over the handshake that provides key confirmation; s0484 finished_key is derived from the Base Key, which is server_handshake_traffic_secret (s0406).	rfc9846:6483, s0063, s0406, s0484
A08	negotiated group and share binding	the named group, the client's encapsulation key and the server's ciphertext that the client accepts: they must be the ones of this very handshake and be committed to by the transcript	main s0026..s0027 require the encoding checks on the encapsulation key and the ciphertext, and main:268 says the key schedule commits to the ML-KEM encapsulation key and the ciphertext through key_share, giving resilience against re-encapsulation attacks.	main:268, s0011, s0018, s0023, s0024, s0025, s0026, s0027, s0111

ID	Name	About	What the documents say	Where
K01	session key and shared secret secrecy	the ML-KEM shared secret and the handshake and application traffic keys derived from it, kept from a network attacker	F.1 'Secrecy of the session keys'; main s0034 inserts the ML-KEM shared secret into the TLS 1.3 key schedule in place of the (EC)DHE secret, and s0534 s0554 derive the traffic keys from it.	rfc9846:6483, s0034, s0535, s0554
C01	forward secrecy from the server signature key	the session keys of a completed handshake, kept from an attacker that obtains the server's long-term signature private key after the handshake is complete	F.1 'Forward secret with respect to long-term keys: If the long-term keying material (in this case the signature keys in certificate-based authentication modes) is compromised after the handshake is complete, this does not compromise the security of the session key'.	rfc9846:6504
C04	session secrecy under KEM key disclosure	the ML-KEM shared secret and traffic keys of a session, kept from an attacker that holds that session's ML-KEM decapsulation key	F.1 says forward secrecy holds only 'as long as the session key itself (and all material that could be used to recreate the session key) has been erased', naming 'private keys corresponding to key shares' among the material that must be erased; main s0032 says the peer holding the decapsulation key sk recovers the encapsulation randomness exactly.	rfc9846:6511, s0032
C02	server authentication under key compromise	the server's assurance about its peer: it must accept only a peer that ran the client side of this handshake, even when the attacker holds the server's long-term signature key	F.1 'Key Compromise Impersonation (KCI) resistance: compromising the long-term secret of one actor should not break that actor's authentication of their peer in the given connection'; client certificate authentication is outside this scope, so the peer authentication the server has is key possession.	rfc9846:6522
K01.10	session key secrecy and agreement	the session keys of a handshake, kept from a network attacker, and the same session keys produced on both sides	F.1 'Establishing the same session keys' and 'Secrecy of the session keys'.	rfc9846:6478, rfc9846:6483
A03.11	server identity and transcript binding	the server identity in the Certificate and the handshake transcript that CertificateVerify signs, as seen by the client when it accepts	s0403 CertificateVerify signs Transcript-Hash(Handshake Context, Certificate) and s0452 says the message 'also provides integrity for the handshake up to this point'; F.1 'Peer Authentication'.	rfc9846:6490, s0403, s0452, s0456
C01.12	forward secrecy after signature key leak	the historical session keys of completed handshakes, kept from an attacker that obtains the server's long-term signature key afterwards; the KEM private key is analysed as a separate exposure	F.1 'Forward secret with respect to long-term keys', and the separate sentence that keys intended to be ephemeral must be erased or they become long-term keys in effect.	rfc9846:6504, rfc9846:6516
X01	application data confidentiality	the application data that the client and the server exchange under the application traffic keys	rfc9846:214 'Confidentiality: Data sent over the channel after establishment is only visible to the endpoints', against an attacker with complete control of the network (rfc9846:223).	rfc9846:214
X02	session secrecy under encapsulation randomness	the ML-KEM shared secret and the traffic keys of one handshake	main:295 warns that the disclosure of the output(s) of an insecure RNG can compromise the state of the RNG and that ML-KEM's encapsulation randomness is an additional place where raw RNG output may be disclosed; main s0032 says the peer holding sk recovers that randomness exactly, so this disclosure is a concrete exposure to ask about.	main:295, s0032

ID	Name	About	What the documents say	Where
X03	independence of handshake and application keys	the application traffic keys, kept from an attacker that holds the handshake traffic keys of a completed handshake	F.1 'Individual session keys produced by a handshake should also be distinct and independent' (rfc9846:6497) and rfc9846:6656 on differentiating HKDF-Expand applications by key and label.	rfc9846:6497

Added from the documents by the analyst: X01, X02, X03.

The security statements of the documents, paragraph by paragraph

The host listed the 8 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: 4. Not taken: 4. 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:253-266	K01, A03, `network`	
P02	main:268-273	A08, A03, `network`	
P03	main:275-278	The prover cannot be asked	It refers implementers to NIST SP 800-227 and recommends side-channel resistant implementations; side channels and implementation quality are outside a symbolic model.
P04	main:280-284	No goal stated	Deployment advice: implementations must weigh their own security, performance and operational constraints and consult the Recommended column of the registry. It states nothing that must hold or can be attacked.
P05 to P06	main:288-292	No goal stated	These two units are the page footer and header of the draft (expiry and page number, Internet-Draft name and date); they state no goal.
P07	main:295-302	X02, `encap_randomness`	
P08	rfc9846:4676-4677	A02, A03, A04, K01, A08, C01, C02, `network`, `server_sig_key`	

The catalogue of properties, entry by entry

The properties were drafted from a catalogue of security and privacy properties. The draft left out 33 of its entries, and the analyst went through each of them: examined here as a property, or left out with the reason. Examined: 10. Left out: 23. 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	X01	
K03	Strong secrecy / indistinguishability of the secret value	K01, X01	
K04	Key usage, direction and session separation	X03	
D01	Message integrity and authenticity of data origin	A03, A08	

Entry	Property of the catalogue	Examined as	Why not
A01	Peer participation (aliveness)	A02	
A05	Implicit key authentication	A03, K01	
A07	Identity–credential–key binding	Outside the scope	Certificate and identity verification are left to the application protocol; in this scope the certificate binds the server name to the signature key by assumption, so a misbinding between credentials cannot arise inside the model.
R01	Authentication / challenge freshness	A04	
R02	Message-level replay resistance	A04	
R03	At most once per business operation	Not applicable	The handshake performs no billable or consumable operation of its own.
R04	Message acceptance order / window safety	Not applicable	The 1-RTT handshake in this scope has no acceptance window, counter or reordering rule; the record layer's own window is outside the modelled part.
R05	Downgrade resistance / security policy preservation	A08	
C03	Known-key security / leak isolation	C01, K01	
C05	Post-compromise security recovery (PCS)	The prover cannot be asked	Post-compromise recovery needs a handshake ordered after the long-term key leak, which cannot be asked together with the earlier handshakes the rest of the property set needs; the documents state the answer already, that TLS gives no security for handshakes after the long-term secret is compromised (rfc9846:6711).
C06	Enhanced authentication after registration credential leak	Not applicable	The protocol has no registration record or password-based credential here: PSK and client certificates are outside the scope.
C07	Resistance to offline dictionary attacks	Not applicable	No password or other low-entropy human secret takes part.
C08	Goal preservation under failure of a specified primitive	The prover cannot be asked	In this scope key establishment is pure ML-KEM with no second component to stand in, and whether a primitive breaks is a computational question a symbolic model does not decide.
Z01	Correctness of authorization enforcement	Not applicable	The handshake grants no access to a resource.
Z02	No expansion of delegated authority	Not applicable	There is no delegation.
Z03	Binding of authorization to identity / session / operation	Not applicable	There is no authorization token to bind.
Z04	Revocation and expiry take effect	Not applicable	There is no revocable or expiring permission inside the handshake.
Z05	Cross-tenant / cross-service security isolation	Not applicable	There is no multi-tenant boundary in the protocol.
S01	Uniqueness of encryption nonces	Outside the scope	AEAD nonces belong to the record layer, which this scope does not model; main s0029 does forbid reusing ML-KEM ciphertexts, and the model gives each handshake a fresh ciphertext.
S02	Safety of retransmission / key reinstallation	Not applicable	TLS 1.3 over a reliable stream has no retransmission and no key reinstallation step in the handshake.

Entry	Property of the catalogue	Examined as	Why not
S03	Rollback resistance and state monotonicity	Not applicable	The handshake keeps no rollback-sensitive version or counter in this scope.
S04	Consistency of the security context across phases	Not applicable	There is no registration, migration or recovery phase to misfit.
S05	Termination and invalidation of old state	Outside the scope	Alert processing and connection teardown are outside the modelled handshake.
P01	Identity anonymity	Outside the scope	Anonymity is not a goal here: TLS 1.3 authenticates the server by certificate and the task scope is a single named server.
P02	Session unlinkability	Outside the scope	Unlinkability between sessions is not a goal of this scope, which fixes one server and no resumption.
P03	Hiding the existence of users / credentials	Not applicable	The protocol has no user or credential database that a response could enumerate.
V01	Conditional progress / error recovery	The prover cannot be asked	ProVerif decides properties of traces, not whether a run always makes progress.
V02	No unexpected deadlock	The prover cannot be asked	Deadlock freedom is not a property a symbolic prover of this kind decides.
V03	Abuse of limited resources / amplification resistance	The prover cannot be asked	Resource abuse and amplification are outside what the traces of this model can say.

The parties and the protected values, one by one

The host listed the 18 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: 5. Without: 13. 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	`client_kem_key`	
S02	Server	party	`server_sig_key`	
S03	Attacker	party	Nothing gained	This is the network attacker itself; it holds nothing beyond what its attacker entries say.
S04	Server identity	value	Nothing gained	The server identity is public by design: the client knows which server it is calling and the certificate is a public value.
S05	Client identity	value	Nothing gained	Client certificate authentication is outside the scope, so the attacker gains nothing from the client identity: no property in the target list turns on it.
S06	ML-KEM shared secret	value	Everything lost by definition	Whoever holds the ML-KEM shared secret of a handshake derives all its traffic keys; that handshake's secrecy is lost by definition and no target asks it under this exposure.
S07	Negotiated parameters	value	Nothing gained	The negotiated parameters, in this scope the fixed ML-KEM-768 group identifier, are public values on the wire.
S08	Handshake transcript	value	Nothing gained	The transcript consists of the messages themselves, which the network attacker already sees; what it cannot supply are the secrets the derivations take.
S09	Server Random	value	Nothing gained	The server Random is sent in the clear in the ServerHello.

ID	Party or value	Kind	Attacker	Why none
S10	Handshake traffic key	value	`handshake_key`	
S11	NamedGroup identifier	value	Nothing gained	The NamedGroup identifier is a registry value sent in the clear.
S12	ML-KEM ciphertext	value	Nothing gained	The ML-KEM ciphertext is the server's public key_share value and travels in the clear in the ServerHello.
S13	Application traffic key	value	Everything lost by definition	Holding an application traffic key decrypts that application data; no target asks a property of a handshake whose application key is already given away.
S14	Historical session key	value	Everything lost by definition	A historical session key gives away that session by definition; the properties that survive are asked under the KEM key and signature key attackers instead.
S15	Server long-term signature private key	value	`server_sig_key`	
S16	ML-KEM decapsulation private key	value	`client_kem_key`	
S17	Session shared secret	value	Everything lost by definition	The shared secret of a handshake gives all of that handshake's traffic keys; nothing in the target list is asked under a party holding it.
S18	Session key	value	Everything lost by definition	A session key that the attacker holds protects nothing; no target asks a property of that handshake under this exposure.

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		
`server_sig_key`	the server's long-term signature private key, as stored on the server, obtained before or during a handshake	F.1 discusses forward secrecy and KCI resistance for exactly this key (rfc9846:6504, rfc9846:6522) and F.1.5 says TLS gives no security for handshakes after it is compromised.	rfc9846:6504, rfc9846:6522, rfc9846:6711
`client_kem_key`	the ML-KEM-768 decapsulation key sk that the client generated for this handshake, read from the client's memory during or after the handshake	F.1 requires the private keys corresponding to key shares to be erased, and warns that failing to do so creates long-term keys in effect (rfc9846:6511); main s0032 says the holder of sk recovers the encapsulation randomness exactly.	rfc9846:6511, s0032
`encap_randomness`	the encapsulation randomness that ML-KEM drew for one handshake, i.e. raw output of the RNG as described in main:295, obtained after the handshake	main:295 says the disclosure of the output(s) of an insecure RNG can be used to compromise the state of that RNG, and that the ML-KEM encapsulation randomness is an additional place where raw RNG output may be disclosed; main s0032 says the peer with sk recovers this randomness exactly, so whoever sees it holds the shared secret.	main:295, s0029, s0032

Attacker	Has	Basis	Where
handshake_key	the handshake traffic keys of one completed handshake (client_handshake_traffic_secret and server_handshake_traffic_secret, or the keys derived from them), obtained after the handshake	F.1 requires handshake secrets to be erased and says the individual session keys of a handshake should be distinct and independent from each other (rfc9846:6495, rfc9846:6511).	rfc9846:6495, rfc9846:6511

ID	network	server_sig_key	client_kem_key	encap_randomness	handshake_key
A02	Holds				
A03	Holds				
A04	Holds				
A06	Holds				
A08	Holds				
K01	Holds				
C01		Holds			
C04			Fails		
C02		Fails			
K01.10	Holds				
A03.11	Holds				
C01.12		Holds			
X01	Holds				
X02				Fails	
X03					Holds

Source and targets

Source: 592 units, 151676 bytes, frozen in source-delivery-01.json. Targets: 9 approved (A02, A03, A04, A06, A08, K01, C01, C04, C02); 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-1701ef07bfcd5a63586c0adfc95e

This step as it reports itself: 1 requests (1 failed), 0.000 USD, 70042 tokens in and 4080 out, 26 s.

Notes on the prover and the model

What the analyst named for targets:

step	targets	features	reason
29	A02, A03, A04, A06, A08, A03.11, K01, K01.10, C01, C01.12, C02, C04, X01, X02, X03	plain_primitives, many_sessions	The model uses only hashing, HKDF (extract, derive-secret, finished key), a MAC, AEAD encryption, a signature and ML-KEM as an abstract encapsulation/decapsulation pair, with no algebraic laws at all: ML-KEM is a KEM whose decapsulation returns the encapsulation randomness, never a group operation. Every target is asked for any number of sessions, and the handshake transcript is modelled as an explicit hash chain rather than as a real transcript (a faithful rendering of Appendix F.1's transcript hash, but an idealisation: the model has no length fields, no record layer and no padding).

The analyst's closing note

Model of TLS 1.3 with pure ML-KEM-768 (server certificate authentication, full 1-RTT), built from main s0021-s0034 and rfc9846 Figure 1, sections 4.5 and 7.1, with a phase-separated compromise ladder (phase 0 handshakes only; phase 1 plus the server's signature key and the handshake traffic keys; phase 2 plus the ephemeral ML-KEM secrets).

Results: all 15 targets have an answer. Proved for any number of sessions: server authentication (A02), agreement on group, shares and shared secret (A03), injective agreement (A04), server Finished key confirmation (A06), binding of the accepted group and key shares (A08), server identity and transcript binding (A03.11), secrecy of the shared secret and traffic keys against the network (K01, K01.10), agreement on the application keys (K01.10), application data confidentiality (X01), forward secrecy after the server signature key leaks (C01, C01.12), and independence of handshake and application keys (X03). Falsified, and every step allowed by the source: the attacker acting as a client is accepted by the server without any client authentication (C02, rfc9846:6485), disclosure of the client's ML-KEM decapsulation key (C04, rfc9846:6511) or of the encapsulation randomness (X02, main:295) gives the whole key schedule. All three failures are stated in the documents, so there is no finding.

Blocked / not covered: ML-KEM is abstract (no implicit rejection, no FIPS 203 encoding or length checks), the transcript is a hash chain rather than the byte-exact hash, the record layer, traffic analysis and post-compromise security are out of the model, and no target rests on a value that changes across steps, so no bounded model or state review applies. Tamarin+Agent would be needed for the algebraic and record-layer parts. Notes in notes.md.