Security analysis of Review TLS hybrid full transcript

This report in another language: 中文.

rfc10024 · ProVerif+ Agent report

Documents	Post-Quantum Traditional (PQ/T) Hybrid Key Agreement Mechanisms for TLS 1.3 (protocol definition); Hybrid Key Exchange in TLS 1.3 (supporting); The Transport Layer Security (TLS) Protocol Version 1.3 (supporting)
Properties examined	12: 8 hold, 4 fail (4 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

The protocol is the TLS 1.3 handshake with the hybrid group X25519MLKEM768: a client and a server agree on an ephemeral hybrid secret, and the server authenticates itself with a certificate. The client holds an ML-KEM-768 key pair, an X25519 ephemeral scalar and a ClientHello random; the server holds its long-term signing key and picks a fresh ML-KEM encapsulation, a fresh X25519 scalar and a ServerHello random.

The client's key_exchange is the encapsulation key followed by the X25519 share, the server answers with an ML-KEM ciphertext followed by its X25519 share, and both concatenate the two component shared secrets into the value the TLS 1.3 key schedule consumes. Because only the public shares enter the transcript hash, every handshake message is bound: the server signs the transcript through its Certificate, and the two Finished messages confirm that both ends derived the same handshake keys from the same hybrid secret.

The parties:

- **Client**: initiates, holds no client certificate. Holds: server certificate (verification key), trusted.
- **Server**: authenticates with a certificate. Holds: signing key and certificate.

Figure 1 shows the exchange as the documents specify it.

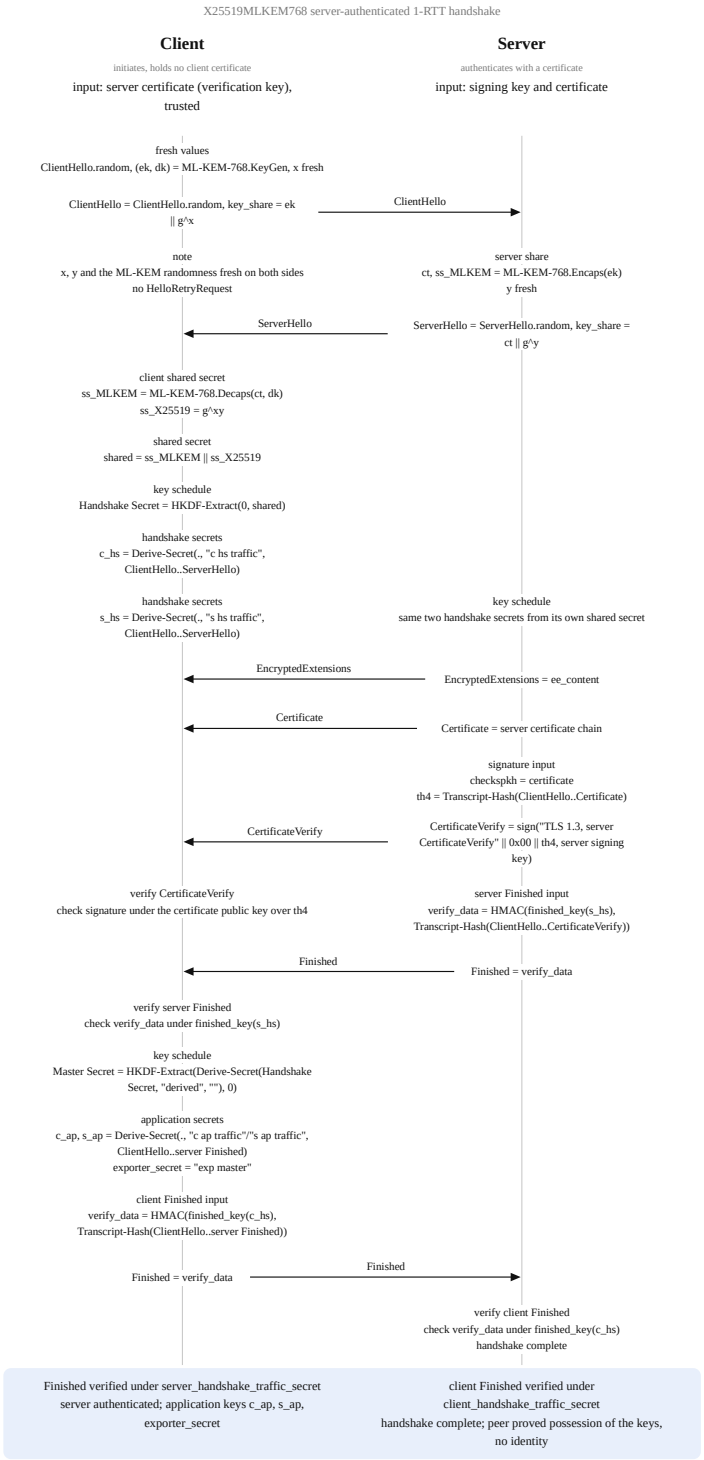


Figure 1. The protocol as specified.

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 — one 1-RTT handshake, server-authenticated, X25519MLKEM768, no HRR, no PSK, no 0-RTT — the protocol gives what the profile is for. Twelve targets were examined; eleven have their answer and the twelfth is the method check, which is met. No new weakness was found: every negative result reproduced here is one the documents already state, so the report has no finding. The guarantees are proofs over a model with an ideal signature, ideal KDF and hashes, and X25519 reduced to the law g^xy = g^yx, so they hold under those assumptions; Tamarin+ Agent, which keeps the exponents and the KEM as they are, is the tool that would remove the second. The one behaviour worth a reader's attention is that a server in this mode cannot tell an honest client from a client session the attacker runs itself, which RFC 9846 states and which the model reproduces.

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.** Whenever a client completes the handshake and accepts the server's identity, there exists a genuine run of the server whose intended peer identity is that client.	no compromise	Holds	medium	yes
A03	**agreement on session keys.** Whenever a party completes the handshake, there exists a matching run of the peer, and both agree on the identities, the negotiated group, the hybrid shared secret, and the handshake/application traffic keys derived from it.	no compromise	Attack holds (stated in the documents)	medium	yes
A04	**uniqueness of acceptances.** Each successful acceptance by a client or a server corresponds to a distinct genuine run of the peer, and the same peer run cannot be matched by two acceptances.	no compromise	Attack holds (stated in the documents)	medium	yes
A06	**Finished key confirmation.** At the point of acceptance, the verifier requires the peer to have genuinely derived and sent a matching Finished confirmation for the same key of this session.	no compromise	Attack holds (stated in the documents)	medium	yes
A08	**transcript and role binding.** The accepted negotiated group, role and transcript correspond to the approved peer proposal and the current session context; CertificateVerify and Finished cover the full message sequence.	no compromise	Holds	medium	yes
K01	**secrecy of session keys.** The attacker cannot derive the server's long-term signing private key, the handshake traffic keys, the application traffic keys or the exporter secret.	no compromise	Holds	medium	yes
C01	**forward secrecy against long-term key leak.** After the session completes and ephemeral state is erased, leaking the server's long-term signing private key still does not let the attacker obtain the handshake or application traffic keys of earlier sessions.	server long-term signing private key (after the session completes and ephemeral secrets are erased)	Holds	medium	yes
C04	**hybrid claim under one broken component.** When either the X25519 ephemeral private key or the ML-KEM decapsulation private key leaks, the attacker still cannot obtain the application traffic keys; each leak combination is recorded separately.	X25519 ephemeral private key or ML-KEM-768 decapsulation private key (single component) (during or after the session)	Holds	medium	yes

ID	Property	Compromise assumed	Result	Value	In the documents
K04	**separation of derived keys.** The keys for the client/server directions, the handshake/application-data purposes and different sessions are all distinct, and the derivation context is bound correctly.	no compromise	Holds	low	yes
A08.10	**CertificateVerify binds full transcript.** CertificateVerify and Finished cover the full handshake message sequence and bind the sender's role; the receiver does not treat evidence from another sequence or another role as valid for this one.	no compromise	Holds	medium	yes
C04.11	**one component broken, keys still secret.** When either the X25519 ephemeral private key or the ML-KEM decapsulation private key leaks, the attacker still cannot obtain the application traffic keys.	X25519 ephemeral private key or ML-KEM-768 decapsulation private key (single component) (during or after the session)	Holds	medium	yes
Q3	**attacks on the baseline model.** The attack trace is required to reproduce in a baseline model that keeps the specification checks.	no compromise	Attack holds (stated in the documents)	low	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).

The questions asked, and the analyst's answers:

Question	Answer	Rests on
Q1: CertificateVerify/Finished binding to the full message sequence and role	Yes on the client side: the CertificateVerify the client accepts covers ClientHello through Certificate with the server context string, and the Finished values chain on from there, so neither can be moved to another sequence or reused by the other role. The server cannot make the same statement about its peer, because the client is not authenticated in this mode and may be a session the attacker drives.	A08, A08.10, A02, A06
Q2: application traffic key secrecy and distinguishing a single KEM/DH component leak	With only the X25519 scalar recovered, or only the ML-KEM decapsulation key known, the application traffic key stays secret; with both, it is lost. That is exactly the hybrid claim, and each direction is a separate answer in the report.	C04, C04.11, K01
Q3: every attack must reproduce in the baseline model with the specification checks left in place	All reported traces come from the single baseline model, which keeps the certificate check, the signature verification and both Finished checks; the only acceptance by a party that the protocol does not authenticate is the server accepting a Finished from the attacker's own client session, which RFC 9846 Appendix F.1 states.	Q3, A03, A06

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:

- Session keys of different sessions are not shown to be distinct from each other; only the key separation inside one session was checked.
- The suite's AEAD, the record layer, nonce handling and the KeyUpdate mechanism are not modelled.

- X25519 is exponentiation with the single law g^xy = g^yx; no other algebraic property and no quantum attacker is modelled. The one-component claims of C04 hold in that abstraction.
- ML-KEM is represented by an encapsulation function and a decapsulation destructor; the encapsulation key check of RFC 10024 section 4.2 and the ciphertext-length check are not modelled.
- Certificate chain validation, revocation and signature algorithm checks are out of scope; the certificate is compared as a value.
- The compromise of a party is modelled only as the three leaks named: the X25519 scalar, the ML-KEM decapsulation key, and the server signing key after the handshake.
- HRR, PSK, 0-RTT, resumption and client certificates are excluded by the scope, so nothing here speaks about them.

What the host records:

- The answers of A02, A03, A04, A06, A08, K01, C01, C04, K04, A08.10, C04.11 rest on an operation the model simplifies (Appendix D).
- Proved without a control query (Appendix C): A02: a02, A03: a02, A04: a04c, A06: a06s, A08: a08, K04: k04_dir, K04: k04_pur, A08.10: a08. A control is a variant of the property that must fail; without one, nothing shows that the property as written can fail in this model.

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

Whenever a client completes the handshake and accepts the server's identity, there exists a genuine run of the server whose intended peer identity is that client.

Every time the client completes the handshake, a server exists that ran the same session: it signed that same transcript and sent a Finished under the handshake key of that session. A network attacker cannot make the client accept a session no server ran, nor move a server's flight into another client session.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value medium: A central authentication guarantee proved over unbounded sessions, but on a model that idealises the signature and restricts the Diffie-Hellman algebra.
- Stated in the documents: yes. Where the documents say it: rfc9846:6490
- Rests on: X25519 is exponentiation with only g^xy = g^yx; the Finished check is where the law is applied, and the session is named by the transcript hash through Certificate, which uses public values only; the certificate check is modelled as an equality on the certificate value; chain validation is out of scope; hashing, the KDF and the signature are ideal

A03: agreement on session keys

Whenever one party completes the handshake, a matching run of the peer exists, and both agree on the identities, the negotiated group, the hybrid shared secret and the handshake/application traffic keys derived from it.

For a session the client completes, both ends hold the same handshake key and the same hybrid shared secret: the server Finished the client accepted is a MAC under that key over that transcript. For the server side nothing can be concluded about the peer, because in this mode only the server is authenticated: a server accepts a Finished from any client session, including one the attacker runs itself.

- Result: Attack holds (stated in the documents); answers what is asked: partly; usable with caveats
- Value medium: The positive half is the useful guarantee; the negative half is the protocol's stated behaviour, so the result is a mechanical confirmation with exact conditions.
- Stated in the documents: yes. Where the documents say it: rfc9846:6486
- Rests on: agreement on key values is read through the Finished check, not through a direct equality of key terms, which ProVerif cannot state here; X25519 algebra limited to g^xy = g^yx; the client has no credential in the model, so the server has nothing to authenticate it with
- Open: agreement on the key values themselves as a query of its own

A04: uniqueness of acceptances

Every successful acceptance by a client or a server corresponds to a distinct real run of the peer, and the same peer run cannot be matched by two acceptances.

Each client acceptance matches a distinct server run, so the same server flight cannot be replayed into two acceptances. A server acceptance does not match a distinct run of an honest client: the attacker can open its own client session with the server, which is a real client run, so no honest session is reused.

- Result: Attack holds (stated in the documents); answers what is asked: partly; usable with caveats
- Value medium: Uniqueness of session keys is confirmed on the client side and the documented gap on the server side is reproduced.
- Stated in the documents: yes. Where the documents say it: rfc9846:6486
- Rests on: the session identifier is the transcript hash through Certificate, so distinctness of runs is distinctness of that hash; unbounded sessions in one run; freshness comes from the ClientHello random and both ephemeral shares

A06: Finished key confirmation

At the point of confirmed acceptance, the verifier requires the peer to have genuinely derived the same key for this session and to have sent a matching Finished confirmation.

The client accepts a server Finished only if it is a MAC under the handshake traffic key derived from the same hybrid shared secret and the same transcript, so it confirms the key and not merely the transcript. The server verifies its client Finished the same way, but cannot tell which party computed it.

- Result: Attack holds (stated in the documents); answers what is asked: partly; usable with caveats
- Value medium: Confirms key confirmation on the client side and gives the exact boundary on the server side.
- Stated in the documents: yes. Where the documents say it: rfc9846:6435
- Rests on: MAC idealisation: the attacker cannot forge a Finished without the handshake key; the server side is stated as a key confirmation, not as authentication of the peer

A08: transcript and role binding

The accepted negotiated group, role and transcript correspond to the approved peer proposal and the current session context, and CertificateVerify and Finished cover the full message sequence.

The CertificateVerify the client accepts is one the server produced over the transcript through Certificate, so the group, both shares, the EncryptedExtensions and the certificate are bound before the signature. A signature from another session or from the client role cannot be moved into this one.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value medium: Proves the transcript binding the RFC 10024 security argument rests on, under an ideal signature.
- Stated in the documents: yes. Where the documents say it: rfc9846:6413, rfc9846:6420
- Rests on: the signature is ideal and unforgeable; the context string separates the server and client role; the client role is not modelled

A08.10: CertificateVerify binds full transcript

CertificateVerify and Finished cover the full handshake message sequence and bind the sender's role, so the receiver does not treat evidence from another sequence or another role as valid for this session.

Answering the question directly: CertificateVerify covers ClientHello, ServerHello, EncryptedExtensions and Certificate, and its context string names the signing role; each Finished covers that transcript plus the messages before it. Neither message can be moved to another sequence or reused by the other role.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value medium: The requested check of full-sequence and role binding is answered positively, with the HRR path excluded by scope.
- Stated in the documents: yes. Where the documents say it: rfc9846:6413
- Rests on: no HRR, so the message_hash substitution of the transcript never applies; no client certificate, so only the server context string occurs

K01: secrecy of session keys

The attacker cannot derive the server's long-term signing private key, the handshake traffic keys, the application traffic keys or the exporter secret.

None of the server signing key, the client and server handshake traffic secrets, the client and server application traffic secrets, or the exporter secret is derivable by an attacker who only has the network. Each control confirms the query can fail: the same key leaks once both ephemeral components of a session are published.

- Result: Holds; answers what is asked: yes; usable
- Value medium: A clean statement of the secrecy goal with controls, over unbounded sessions; the simplification of the DH algebra does not help the attacker here.
- Stated in the documents: yes. Where the documents say it: rfc9846:6483
- Rests on: the AEAD and the KDF are ideal; secrecy is asked of the secrets, not of the traffic keys derived from them; the client keys are asked only for sessions in which the server also ran

C01: forward secrecy against long-term key leak

After a session completes and the ephemeral state is erased, leaking the server's long-term signing private key still does not let the attacker obtain the handshake or application traffic keys of earlier sessions.

Once the handshake is finished and the ephemeral state is gone, publishing the server's long-term signing key does not give the application traffic secrets of the sessions the server ran: those depend on the per-session hybrid secret only. The control shows that the same query fails as soon as both ephemeral components of a session are published.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value medium: Forward secrecy with respect to the long-term key is a central claim of this profile and is shown to depend only on the ephemeral components staying secret.
- Stated in the documents: yes. Where the documents say it: rfc9846:6504
- Rests on: the leak happens after the handshake; the model has no erasure failure, so it assumes the ephemeral secrets are not kept; X25519 algebra limited to g^xy = g^yx; the model cannot say that the ephemeral scalars resist a quantum attacker

C04: hybrid claim under one broken component

When one of the X25519 ephemeral private key or the ML-KEM decapsulation private key is leaked, the attacker still cannot obtain the application traffic keys; each leak combination is recorded separately.

Recovering the X25519 scalar of a session, as a passive observer with a quantum computer could, leaves the application traffic secrets secret as long as the ML-KEM decapsulation key is kept: the ML-KEM component of the concatenated secret is still unknown. Knowing the ML-KEM decapsulation key alone is symmetric: the X25519 component still protects. Publishing both components breaks the session, as the control shows.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value medium: This is the property the hybrid profile exists for, confirmed per component; the cryptography of the combiner itself remains an assumption.
- Stated in the documents: yes. Where the documents say it: main:299, rfc9954:550
- Rests on: the hybrid secret is the concatenation of the two component secrets entering HKDF; the dual-PRF argument of RFC 9954 is not modelled, only the idealisation of the KDF; ML-KEM is a destructor-based encapsulation, so the ML-KEM shared secret is secret unless the decapsulation key is known; the X25519 scalar is published as a value, standing for recovery by a quantum attacker

C04.11: one component broken, keys still secret

When one of the X25519 ephemeral private key or the ML-KEM decapsulation private key is leaked, the attacker still cannot obtain the application traffic keys.

To answer the question directly: with only one of the two ephemeral components broken, the application traffic key of that session is not derivable; with both broken it is. The two directions are answered by separate queries, so the report shows which component carries the guarantee.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value medium: Directly answers the practical question the profile is specified for, one component at a time.
- Stated in the documents: yes. Where the documents say it: rfc9954:550
- Rests on: ideal KDF and MAC; the concatenation of fixed-length secrets is injective in the model; no claim about a quantum attacker breaking the combiner itself

K04: separation of derived keys

The keys for the client and server directions, for the handshake and application data purposes, and for different sessions are all distinct, and the derivation context is bound correctly.

The client and server application traffic secrets of one session are different values, and the exporter secret differs from the application secrets, because each is derived with its own label from the main secret. A key of one direction or one purpose is therefore never usable as another.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value low: Follows directly from the label discipline in the key schedule; worth recording as a check, not as a result.
- Stated in the documents: yes. Where the documents say it: rfc9846:6495
- Rests on: labels are constants and the KDF is ideal, so distinct labels give distinct outputs; not covered: distinctness between two different sessions' secrets, and the traffic keys derived from a secret

Q3: attacks on the baseline model

The requirement is that attack traces be reproduced in the baseline model that keeps the specification checks.

Every trace the analysis reports comes from the one model, which keeps the certificate equality check, the CertificateVerify signature verification and both Finished verifications. The trace recorded for this target is the server accepting a Finished from a client session the attacker drives; it is reachable with every check in place, and it is the behaviour RFC 9846 Appendix F.1 states for unilaterally authenticated connections.

- Result: Attack holds (stated in the documents); answers what is asked: yes; usable

- Value low: This is a requirement of the analysis, not a statement of the documents; its answer is that the requirement is met and the one trace it names is the documented one.
- Stated in the documents: yes. Where the documents say it: rfc9846:6486
- Rests on: the model is the baseline: no branch removes a check; a complete handshake exists on that model, which is what was checked

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			1 proved
A03	proved		a02		a06c (holds_in_source)	1 proved, 1 attack
A04	proved		a04c		a04s (holds_in_source)	1 proved, 1 attack
A06	proved		a06s		a06c (holds_in_source)	1 proved, 1 attack
A08	proved		a08			1 proved
K01	proved		k01_ssk	checked		6 proved
C01	proved		c01	checked		1 proved
C04	proved		c04_x	checked		2 proved
K04	proved		k04_dir			2 proved
A08.10	proved		a08			1 proved
C04.11	proved		c04_x	checked		2 proved
Q3	reachable		run		a06c (holds_in_source)	2 reachable, 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	Whenever the client completes the handshake, a server run exists for the same transcript: it sent Certificate, CertificateVerify and a server Finished over that same message sequence.	`network`	proved	a02	none
A03	The client accepts only a transcript a server ran, and since the server Finished it checked was a MAC under its own handshake traffic key, both endpoints hold the same handshake key and the same hybrid shared secret.	`network`	proved	a02	none
A03	Whenever the server completes the handshake, the peer it accepted is a client run of that same session.	`network`	attack (holds_in_source)	a06c	
A04	Every client acceptance corresponds to a distinct server run.	`network`	proved	a04c	none
A04	Every server acceptance corresponds to a distinct client run of that same session.	`network`	attack (holds_in_source)	a04s	
A06	The client accepts a server Finished only if it is a MAC under the server handshake traffic key derived from the same hybrid shared secret and the same transcript.	`network`	proved	a06s	none

target	claim	attacker	answer	query	control
A06	The server accepts a client Finished only if it is a MAC under the client handshake traffic key it derived.	`network`	attack (holds_in_source)	a06c	
A08	Every CertificateVerify the client accepts was produced by the server over the transcript through Certificate, including the negotiated group, the shares and the identities carried in the chain.	`network`	proved	a08	none
K01	The server long-term signing private key is not derivable by a network attacker.	`network`	proved	k01_ssk	checked
K01	The client application traffic secret of a session is not derivable by a network attacker.	`network`	proved	k01_cap	checked
K01	The server application traffic secret of a session is not derivable by a network attacker.	`network`	proved	k01_sap	checked
K01	The client handshake traffic secret of a session is not derivable by a network attacker.	`network`	proved	k01_chs	checked
K01	The server handshake traffic secret of a session is not derivable by a network attacker.	`network`	proved	k01_shs	checked
K01	The exporter secret of a session is not derivable by a network attacker.	`network`	proved	k01_exp	checked
C01	After the server long-term signing key is compromised and the ephemeral state erased, the application traffic secrets of the sessions a genuine server ran stay secret.	`server_sig_key_after_session`	proved	c01	checked
C04	With the session's X25519 scalar recovered and the ML-KEM decapsulation key still secret, the application traffic secrets of that session stay secret.	`x25519_scalar_crqc`	proved	c04_x	checked
C04	With the client's ML-KEM-768 decapsulation key known and the X25519 scalar still secret, the application traffic secrets of that session stay secret.	`mlkem_decaps_key`	proved	c04_k	checked
K04	The client and server application traffic secrets of the same session are distinct.	`network`	proved	k04_dir	none
K04	The exporter secret of a session is distinct from its client application traffic secret, each key being derived with its own label from the main secret.	`network`	proved	k04_pur	none
A08.10	CertificateVerify covers the whole sequence ClientHello, ServerHello, EncryptedExtensions, Certificate and binds the server role through its context string; the Finished values then cover that plus the signature, so neither can be moved to another session or used by the other role.	`network`	proved	a08	none

target	claim	attacker	answer	query	control
C04.11	With the X25519 ephemeral scalar recovered on its own, the application traffic key of the session stays secret.	`x25519_scalar_crqc`	proved	c04_x	checked
C04.11	With the ML-KEM-768 decapsulation key known on its own, the application traffic key of the session stays secret.	`mlkem_decaps_key`	proved	c04_k	checked
Q3	Every attack reported is a trace of the baseline model, which keeps the certificate equality check, the CertificateVerify signature verification and both Finished checks, so no check was removed to obtain it.		reachable	run	
Q3	Every attack trace reported is a trace of the baseline model, which keeps the certificate equality check, the CertificateVerify signature verification and both Finished checks, so no check of the specification was removed.	`network`	reachable	run	
Q3	The trace the analysis reports is reachable on the baseline model, which keeps the certificate equality check, the CertificateVerify signature verification and both Finished checks: no specification check was removed to obtain it.	`network`	attack (holds_in_source)	a06c	

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-20261114T233005, 2026-11-14. Status: finished. Model: deepseek-v4-1-flash-261017. Steps: 58, prover 109 s, wall 15 min, cost 0.207 USD.

Protocol	Review TLS hybrid full transcript
Version	rfc10024
Analysed	A simplified, independently corrected model of the first-round hello-only transcript: fixed X25519MLKEM768 server authentication, 1-RTT, all handshake messages enter the corresponding transcript, CertificateVerify, server/client Finished, handshake message AEAD and application/exporter secrets are layered per the baseline, and key control cannot be replaced by public certificates/Finished. HRR/PSK/0-RTT are excluded.
Not analysed	The symbolic model does not prove computational security or post-quantum security of the algorithms; attack traces hold only under explicit assumptions and are not directly equivalent to specification vulnerabilities. Every property must have a reachability/positive control.
Source text	549 numbered passages (144873 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	4	Negotiated Groups
main	4.1	Client Share
main	4.2	Server Share
main	4.3	Shared Secret
main	6	Security Considerations
rfc9954	3	Construction for Hybrid Key Exchange
rfc9954	3.1	Negotiation
rfc9954	3.2	Transmitting Public Keys and Ciphertexts
rfc9954	3.3	Shared Secret Calculation
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

Document	Section	Title
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
rfc9846	7.4	Asymmetric Shared Secret Calculation
rfc9846	7.4.1	Finite Field Diffie-Hellman
rfc9846	7.4.2	Elliptic Curve Diffie-Hellman

Document	Section	Title
rfc9846	7.5	Exporters

The purpose of the protocol as the source step located it in the documents: A simplified, independently corrected model of the first-round hello-only transcript: fixed X25519MLKEM768 server authentication, 1-RTT, all handshake messages enter the corresponding transcript, CertificateVerify, server/client Finished, handshake message AEAD and application/exporter secrets are layered per the baseline, and key control cannot be replaced by public certificates/Finished. 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	server authentication to the client	The client's assurance that the server it accepted really ran the handshake on the transcript the client saw.	RFC 9846 states peer authentication as a handshake goal: the client's view of the peer identity should reflect the server's identity, and the signature-based modes authenticate the server's signature over the handshake transcript.	rfc9846:6490, rfc9846:6537, s0409, s0435
A03	agreement on session keys	Both endpoints deriving the same session keys and the same negotiated parameters from the same hybrid shared secret.	RFC 9846 lists establishing the same session keys and secrecy of the session keys among the goals an active network attacker must not break.	rfc9846:6478, rfc9846:6483, s0015, s0491
A04	uniqueness of acceptances	Distinct handshakes of the client and of the server producing distinct acceptances, one peer run per acceptance.	RFC 9846 states uniqueness of the session keys: any two distinct handshakes should produce distinct, unrelated session keys.	rfc9846:6495, s0435, s0491
A06	Finished key confirmation	The verifying party requiring a Finished MAC under the handshake key it derived for this session.	RFC 9846 says the Finished message is essential for providing authentication of the handshake and of the computed keys, and recipients must verify its contents.	rfc9846:6435, rfc9846:6436, s0435, s0445
A08	transcript and role binding	The negotiated group, the party roles and the whole message sequence being bound into CertificateVerify and Finished.	RFC 10024 says the security analysis relies crucially on the TLS 1.3 message transcript; RFC 9846 defines the transcript hash over every handshake message and the role context strings.	main:270, rfc9846:6413, rfc9846:6420, s0063, s0068, s0420
K01	secrecy of session keys	The server's long-term signing key, the handshake traffic keys, the application traffic keys and the exporter secret, kept from a network attacker.	RFC 9846 states secrecy of the session keys: the shared session keys should be known only to the communicating parties and not to the attacker.	rfc9846:6483, rfc9846:6511, s0491, s0511, s0546
C01	forward secrecy against long-term key leak	Sessions that a genuine server ran, after the server's long-term signing key is compromised and the ephemeral state erased.	RFC 9846 states forward secrecy with respect to long-term keys: compromising the signature key after the handshake does not compromise the session key, as long as ephemeral material is erased.	rfc9846:6504, s0504, s0511, s0521

ID	Name	About	What the documents say	Where
C04	hybrid claim under one broken component	The application traffic key of a session whose X25519 component is broken or whose ML-KEM decapsulation key is broken, one at a time.	RFC 9954 says the hybrid shared secret is secure as long as at least one component key exchange algorithm is unbroken; RFC 10024 points at the CRQC recovery of the X25519 scalar and at the exposure of the encapsulation randomness.	rfc9954:550, main:299, main:282, s0025, s0051, s0511
K04	separation of derived keys	The client and server keys of the same session, and the application, handshake and exporter keys of one side, being distinct.	RFC 9846 states that individual session keys produced by a handshake should be distinct and independent, and derives each key with its own label and direction.	rfc9846:6495, rfc9846:6497, s0511, s0528
A08.10	CertificateVerify binds full transcript	The same binding, read as the user's question about CertificateVerify and Finished covering the full sequence and the sender's role.	Transcript hash over ClientHello through the authentication messages, and the separate server and client CertificateVerify context strings.	rfc9846:6413, rfc9846:6445, s0063, s0360, s0445
C04.11	one component broken, keys still secret	The same single-component leak read as the user's question: application traffic keys stay secret when only one KEM or DH component is broken.	The purpose of the hybrid construction is that the final session key is secure as long as at least one component algorithm remains unbroken.	rfc9954:550, rfc9954:117, s0015, s0051
Q3	attacks on the baseline model	The analysis method: every attack reported is reproduced on a model that keeps the checks the specification requires.	The task states each attack trace holds only under explicit assumptions and must be reproducible in a baseline model that keeps the specification checks; this is a requirement on the analysis, not a protocol property.	s0019

The analyst added no property: the drafted list covers what the documents state.

The security statements of the documents, paragraph by paragraph

The host listed the 20 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: 13. Not taken: 7. 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:268-271	A08, A08.10, A03	
P02	main:273-276	The prover cannot be asked	Guidance on side-channel resistant implementations; the strength of an implementation is not a trace property.
P03	main:278-280	A03	
P04	main:282-287	C04, `mlkem_decaps_key`	
P05	main:289-297	C04, `mlkem_decaps_key`	
P06	main:299-306	C04, C04.11, `x25519_scalar_crqc`	
P07	main:308-310	C04.11, `x25519_scalar_crqc`, `mlkem_decaps_key`	
P08	rfc7748:783-793	The prover cannot be asked	The security level of curve25519 is a primitive strength, not a trace property.
P09	rfc7748:802-808	The prover cannot be asked	Same: primitive strength of curve448.

Paragraphs	Where	Taken up as	Why not
P10	rfc7748:810-818	A03, `network`	
P11	rfc7748:820-825	A08, `network`	
P12	rfc7748:827-837	Outside the scope	Observable differences between curve implementations are an implementation fingerprinting issue outside the modelled handshake.
P13	rfc9846:4676-4677	A02, A03, A04, A06, A08, A08.10, K01, K04, C01, `network`, `server_sig_key_after_session`	
P14	rfc9954:550-558	C04, C04.11, `x25519_scalar_crqc`, `mlkem_decaps_key`	
P15	rfc9954:560-569	The prover cannot be asked	IND-CCA2 security of the KEM and security under public key reuse are computational assumptions, not trace properties.
P16	rfc9954:571-574	A03	
P17	rfc9954:576-582	The prover cannot be asked	Timing side channels of variable-length secrets are not askable.
P18	rfc9954:584-586	The prover cannot be asked	Collision resistance of the hash used by the KDF is a computational assumption.
P19	rfc9954:588-592	A03	
P20	rfc9954:594-597	A03	

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: 9. Left out: 24. 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	K01	
D01	Message integrity and data origin authenticity	A08, A06	
A01	Peer aliveness	A02, A03	
A05	Implicit key authentication	A03, K01	
A07	Identity–credential–key binding	A08	
R01	Authentication / challenge freshness	A04	
R02	Message-level anti-replay	Outside the scope	Record-layer replay windows are not modelled; the task covers the handshake only.
R03	At-most-once business operation	Not applicable	The protocol carries no business operations.
R04	Message acceptance order / window safety	Outside the scope	Acceptance windows and ordering are record-layer mechanisms outside the modelled handshake.

Entry	Property of the catalogue	Examined as	Why not
R05	Resistance to downgrade / preservation of security policy	Outside the scope	The task fixes the X25519MLKEM768 group and excludes HRR, PSK and 0-RTT, so no negotiation alternative exists in the model.
C02	Resistance to impersonation after key compromise (KCI)	Not applicable	The scope is server-only certificate authentication; the client holds no long-term credential, so there is no peer whose stolen credential could impersonate another peer.
C03	Known-key security / compromise isolation	C04	
C05	Security recovery after compromise (PCS)	Outside the scope	No resumption, PSK or key update is modelled.
C06	Enhanced authentication after compromise of the registration credential	Not applicable	No password or registration record exists in this protocol.
C07	Resistance to offline dictionary attacks	Not applicable	No low-entropy secret is used.
C08	Preservation of goals under failure of the specified primitives	C04, C04.11	
Z01	Correctness of authorisation enforcement	Not applicable	No authorisation layer is part of the handshake.
Z02	No expansion of delegated authority	Not applicable	No delegation.
Z03	Binding of authorisation to identity / session / operation	Not applicable	No authorisation token.
Z04	Revocation and expiry take effect	Not applicable	No revocation or expiry.
Z05	Secure isolation across tenants / across services	Not applicable	No tenants or shared nodes.
S01	Uniqueness of the encryption nonce	Outside the scope	AEAD nonce management of the record layer is not modelled.
S02	Safety under retransmission / key reinstallation	Outside the scope	Key reinstallation and retransmission handling are outside the modelled 1-RTT handshake.
S03	Resistance to rollback and state monotonicity	Outside the scope	The handshake keeps no rollback-sensitive version or counter state in this scope.
S04	Consistency of the security context across phases	Not applicable	The protocol has a single handshake context, with no resumption, migration or registration phase.
S05	Termination and invalidation of old state	Outside the scope	Connection teardown is not modelled.
P01	Identity anonymity	Not applicable	The server identity is public and the client is not authenticated, so there is no anonymity goal.
P02	Session unlinkability	Not applicable	No client identity or ticket is exchanged, so runs cannot be linked by identity.
P03	Hiding of user / credential existence	Not applicable	No user or credential lookup exists.
V01	Conditional progress / error recovery	The prover cannot be asked	Progress under conditions is not a property of traces ProVerif can be asked.
V02	No unexpected deadlock	The prover cannot be asked	Absence of deadlock is not askable.
V03	Abuse of limited resources / resistance to amplification	The prover cannot be asked	Resource consumption beyond counting events is not askable.

The parties and the protected values, one by one

The host listed the 21 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: 2. Without: 19. 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	A client controls only its own ephemeral secrets, and any attacker can already run a client session itself on the network; no property about other sessions changes.
S02	Server	party	`server_sig_key_after_session`	
S03	Server identity	value	Nothing gained	The server certificate and its public key are public.
S04	Client identity	value	Nothing gained	The client is not authenticated in this mode and has no identity value in the protocol.
S05	X25519MLKEM768 hybrid shared secret	value	Everything lost by definition	Every key of that session is a function of the hybrid shared secret; no document asks for secrecy once the attacker holds it.
S06	Handshake traffic key	value	Everything lost by definition	All handshake and application keys of the session derive from the handshake traffic secret.
S07	Application traffic key	value	Everything lost by definition	Holding the application traffic key of a session loses that session's confidentiality by definition.
S08	Negotiated parameters	value	Nothing gained	Negotiated parameters travel in the handshake and are public.
S09	Handshake transcript	value	Nothing gained	The transcript is built from messages the network attacker sees.
S10	Randomness	value	Nothing gained	The hello randoms are public; the ephemeral secrets drawn from the generator are covered separately by the two component-leak attackers.
S11	Session key	value	Everything lost by definition	Holding a session key loses that session by definition.
S12	Finished verify data	value	Nothing gained	Finished verify data is a MAC under a derived key; holding it gives no key.
S13	Negotiated group X25519MLKEM768	value	Nothing gained	The group identifier is public and fixed.
S14	Transcript hash	value	Nothing gained	The transcript hash is computable from public messages.
S15	Role context string	value	Nothing gained	The context strings are public constants.
S16	Server long-term signing private key	value	`server_sig_key_after_session`	
S17	Exporter secret	value	Everything lost by definition	The exporter secret is derived from the same master secret as the traffic keys; holding it loses that session's exporter output by definition.
S18	Handshake traffic key of a past session	value	Everything lost by definition	Holding a session's handshake traffic key loses that session by definition; other sessions use fresh ephemeral secrets.
S19	Application traffic key of a past session	value	Everything lost by definition	Holding a session's application traffic key loses that session by definition.
S20	Hybrid shared secret	value	Everything lost by definition	Same value as S05 for the sessions that use it.

ID	Party or value	Kind	Attacker	Why none
S21	CertificateVerify signature	value	Nothing gained	The CertificateVerify signature is sent in the handshake; it is a signature under the long-term key, not a key.

The properties under each attacker

The attackers are the analyst's own, fixed 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		
`x25519_scalar_crqc`	The X25519 ephemeral private scalars of a completed session (client and server), recovered after the session by a cryptographically relevant quantum computer; the ML-KEM decapsulation key stays secret.	The document warns that a passive observer with a CRQC can recover the ECDH ephemeral scalar from the public share.	main:299
`mlkem_decaps_key`	The ML-KEM-768 decapsulation key the client holds (and so the encapsulation randomness m and the ML-KEM shared secret of the session); the X25519 scalar stays secret.	The document says the client recovers m exactly during decapsulation, and that RNG output disclosed through m is an additional exposure.	main:282, main:289
`server_sig_key_after_session`	The server's long-term signing private key, obtained after the handshake completed and the ephemeral state was erased.	RFC 9846 states forward secrecy with respect to long-term keys: a signature key compromised after the handshake must not compromise the session key.	rfc9846:6504

ID	`network`	`x25519_scalar_crqc`	`mlkem_decaps_key`	`server_sig_key_after_session`
A02	Holds			
A03	Fails			
A04	Fails			
A06	Fails			
A08	Holds			
K01	Holds			
C01				Holds
C04		Holds	Holds	
K04	Holds			
A08.10	Holds			
C04.11		Holds	Holds	
Q3	Fails			

Source and targets

Source: 549 units, 144873 bytes, frozen in source-delivery-01.json. Targets: 9 approved (A02, A03, A04, A06, A08, K01, C01, C04, K04); 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-9ed9cece2ceac30bf32cd9a216a4

This step as it reports itself: 1 requests (1 failed), 0.000 USD, 67700 tokens in and 4653 out, 27 s.

Notes on the prover and the model

- A02: answered in ProVerif, and the analyst names dh_algebra for this target
- A03: answered in ProVerif, and the analyst names dh_algebra for this target
- A04: answered in ProVerif, and the analyst names dh_algebra for this target
- A06: answered in ProVerif, and the analyst names dh_algebra for this target
- A08: answered in ProVerif, and the analyst names dh_algebra for this target
- K01: answered in ProVerif, and the analyst names dh_algebra for this target
- C01: answered in ProVerif, and the analyst names dh_algebra for this target
- C04: answered in ProVerif, and the analyst names dh_algebra for this target
- K04: answered in ProVerif, and the analyst names dh_algebra for this target
- A08.10: answered in ProVerif, and the analyst names dh_algebra for this target
- C04.11: answered in ProVerif, and the analyst names dh_algebra for this target

A ProVerif model that simplifies algebra (xor, dh_algebra, other_algebra) gives the attacker less than the real operation: a proof there is a proof against a weaker attacker, an attack there is an attack. A ProVerif model that leaves out a check on a value that changes (changing_state) lets parties accept more: a proof there holds, an attack there may be nothing but the missing check. Tamarin+ Agent models these as they are.

What the analyst named for targets:

step	targets	features	reason
44	A02, A03, A04, A06, A08, A08.10, K01, C01, C04, C04.11, K04	plain_primitives, dh_algebra, many_sessions	Hashing, MAC, HKDF-style derivation, the AEAD-free key schedule and the certificate signature are modelled as ideal primitives. X25519 is modelled by exponentiation with the single law g^xy = g^yx and no other algebraic property; ProVerif applies that law when it checks a Finished or a signature, so agreement on a key term is only visible through the transcript identifier, and the component-leak results rest on that simplification. Every property is asked over any number of client and server sessions in one run, with the client keys and the server signing key leaked only after the handshake.

The analyst's closing note

Twenty-four answers are recorded on model.pv for the twelve targets. Proved: server authentication to the client and its distinctness (A02, A04), agreement on the session transcript and the handshake key through the Finished check (A03 first claim), client-side key confirmation (A06 first claim), the CertificateVerify transcript and role binding (A08, A08.10), secrecy of the signing key, of both handshake and both application traffic secrets and of the exporter secret (K01, six claims, each with a control), forward secrecy after the server signing key leaks (C01, with control), the hybrid claim one component at a time in both directions (C04, C04.11, with controls), and key separation by direction and purpose (K04). Falsified and recorded as what the documents state: a server in this server-only mode

cannot tell an honest client from a client session the attacker runs itself (A03, A04 and A06 second claims; RFC 9846 Appendix F.1), reproduced on the baseline model that keeps every check. No weakness the documents do not state was found, so the report has no finding. The proofs rest on an ideal signature, KDF and hashes and on X25519 reduced to g^xy = g^yx; no value that changes across steps is in scope, so there is no bounded model and no state review. Tamarin+ Agent, which keeps the exponents and the KEM as they are, is what would remove the algebra restriction from the one-component claims.