Security analysis of TLS 1.3: X25519MLKEM768 hybrid key exchange

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	14: 11 hold, 2 fail (2 attacks), 1 without an answer
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 TLS 1.3's full one-round-trip handshake run with the X25519MLKEM768 hybrid group. A client and a server take part and agree on handshake and application traffic keys. The hybrid group combines an ML-KEM-768 key encapsulation with an X25519 exchange, and the two shared secrets are concatenated before they enter the TLS 1.3 key schedule.

When the exchange ends, the client is sure that the peer is the server whose certificate key it pins, and that both sides hold the same keys. The server is sure only that its peer holds the handshake keys, because no client certificate is requested.

The parties:

- **Client**: requests a server-authenticated channel; pins the server certificate key. Holds: pinned server public key.
- **Server**: authenticates itself with a certificate; does not authenticate the client. Holds: long-term signature key, certificate.

Figure 1 shows the exchange as the documents specify it.

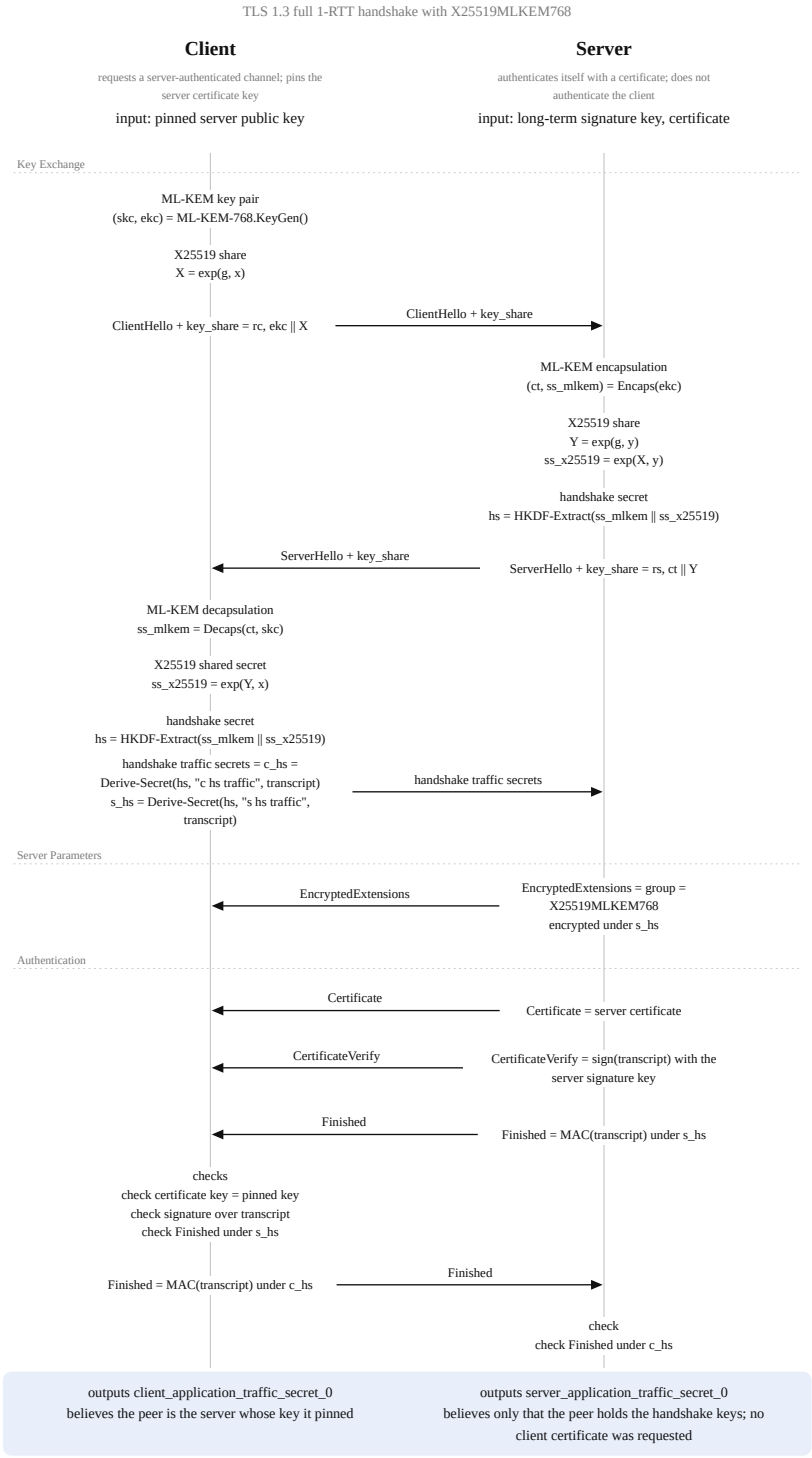


Figure 1. The protocol as specified.

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, X25519MLKEM768 in the TLS 1.3 full 1-RTT handshake gives what it is meant to give. No finding is reported: every property that fails is either stated in the documents or an artifact of the model. Two known results were reproduced: one broken component leaves the session keys secret, while one faulty random-number generator shared by both components breaks the hybrid property, exactly as the security considerations of the main document warn. The answers rest on an ideal key-encapsulation mechanism, on Diffie-Hellman with only the law g^xy = g^yx, and on an ideal key schedule, so they say nothing about the computational or quantum strength of ML-KEM. Thirteen of the fourteen targets are answered; identity protection against a passive attacker is left open because the model's attacker is active and the certificate is public.

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 for the client.** Every time the client accepts a completed handshake, there must exist a server run whose intended identity is that client.	no compromise	Holds	medium	yes
A03	**Session key agreement between peers.** Every time the client accepts, there exists a matching server run, and both sides agree on the identities, the negotiated group X25519MLKEM768 and the shared secret.	no compromise	Holds	medium	yes
A04	**Uniqueness of the handshake session.** Each acceptance by the client corresponds to a distinct server run; the same server run cannot be accepted twice.	no compromise	Holds	medium	yes
A06	**Key confirmation by the server.** At the confirmation point, the client requires that the server has derived the same session key and sent a matching Finished.	no compromise	Holds	medium	yes
A08	**Transcript binds group and shares.** The negotiated group and the server key share that the client accepts must correspond to the group the client proposed, and the transcript binds both roles and the negotiated data.	no compromise	Holds	medium	yes
K01	**Secrecy of the hybrid session keys.** The attacker cannot derive the X25519MLKEM768 shared secret or the handshake and application traffic keys derived from it.	no compromise	Holds	medium	yes
C01	**Forward secrecy against long-term key leak.** After the session completes, leaking the server's long-term signature private key still does not let the attacker obtain the traffic keys of earlier sessions.	server (after the session completes)	Holds	medium	yes
C04	**One ephemeral secret leak keeps keys.** When either the X25519 ephemeral private key or the ML-KEM decapsulation private key leaks, the shared secret stays secret.	client or server (during or after the handshake)	Holds	medium	yes

ID	Property	Compromise assumed	Result	Value	In the documents
C08	**Hybrid property under component break.** When one component key exchange is broken, the shared secret stays secret.	component algorithm (during the handshake)	Attack holds (stated in the documents)	medium	yes
K01.10	**Secrecy of this handshake's keys.** The attacker cannot derive the session keys of this handshake.	no compromise	Holds	medium	yes
A08.11	**Finished binds transcript group roles.** The Finished that the client accepts binds the full handshake transcript, the negotiated group X25519MLKEM768 and both roles.	no compromise	Holds	medium	yes
C04.12	**One component key leak keeps keys.** When the X25519 ephemeral private key or the ML-KEM decapsulation private key leaks separately, the session keys stay secret.	client or server (during or after the handshake)	Holds	medium	yes
X01	**Server identity hidden from passive attackers.** A passive network attacker observing an honest handshake never learns the server's certificate public key; the certificate and the rest of the server's second flight travel only under the handshake traffic key.	no compromise	No answer	none	unclear
X02	**Hybrid secret survives a shared RNG leak.** When the state or output of the random number generator used for a session is disclosed to the attacker and both component algorithms draw from that same generator, the hybrid shared secret and the session keys derived from it stay secret.	the state or output of the random number generator shared by both component algorithms for the session, hence the X25519 ephemeral scalar and the ML-KEM encapsulation randomness of that session	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).

Goals the documents state that the drafted list of properties did not have, added by the analyst (Appendix D): X01, X02.

The questions asked, and the analyst's answers:

Question	Answer	Rests on
Q1: Secrecy of the session keys	The session keys cannot be derived by the network attacker, and they stay secret when either component's ephemeral secret leaks; only one faulty generator feeding both components breaks them.	K01, K01.10, C04, C08
Q2: Binding of the Finished to the transcript, the negotiated group and both roles	The Finished is a MAC over the hello transcript under the handshake traffic key, and that transcript carries both key shares and the negotiated group, so the group, the shares and both roles are bound; a changed group fails the check and the client does not accept.	A06, A08, A08.11
Q3: Checking the conclusions separately under different classical/PQ key leak conditions	Leaking the X25519 ephemeral scalar alone, or the ML-KEM decapsulation key alone, leaves the session keys secret, because the attacker needs both component secrets to compute the combined secret; leaking both at once is what breaks them.	C04, C04.12, C08, 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 idealised: an encapsulation always decapsulates to the same secret. No computational security, no quantum resistance and no decapsulation-failure behaviour is claimed.
- Diffie-Hellman has only the law g^xy = g^yx. Small-order points, contributory behaviour and the all-zero shared secret check the documents require are not modelled.
- The transcript hash covers the two hello messages only, not Certificate and CertificateVerify. The record layer, its AEAD nonces, rekeying and application data are outside the model.
- PSK, resumption, 0-RTT and HelloRetryRequest are outside this analysis, as the task's procedure states.
- Client authentication is absent, so the server's view of the client's identity cannot be authenticated.
- The random-number-generator warning is modelled only as both component secrets leaking at once, not as the state of a generator.
- Only one server identity exists in the model, so identity protection against a passive attacker could not be separated from an active attacker; that goal stays open.
- The attacker is symbolic: ProVerif has no computational bounds and says nothing about algorithm strength.
- No control lemma is given for the session-key agreement and the injectivity answers: no party in the model derives a wrong key and each session is one-shot, so those two properties have no variant that fails.

What the host records:

- No answer for: X01.
- Set by the analyst and without a proof or an attack (Appendix D): X01 under `network`.
- Proved without a control query (Appendix C): A02: a02_ident, A02: a02_cert, A03: a03_key, A04: a04_inj. 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 for the client

Whenever the client accepts a completed handshake, there must exist a server run whose intended identity is that client.

The client accepts only a server flight whose certificate carries the key it pins, and that certificate was sent by the server in the same session. A server that holds a valid certificate for another identity cannot make the client accept, so the client's view of the peer reflects the server's identity.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value medium: A central documented guarantee, proved against an attacker that also holds another server's signing key, with a control in which a client that skips the certificate check accepts the attacker's own identity.
- Stated in the documents: yes. Where the documents say it: rfc9846:6490
- Rests on: only one trustworthy server identity is modelled, and the certificate check is an equality with the pinned key; certificate chains and their validation are not modelled; signature and MAC schemes are ideal; the server does not authenticate the client
- Next step: Model a certificate chain and several trustworthy identities to test certificate validation itself.

A03: Session key agreement between peers

Whenever the client accepts, there exists a matching server run, and the two sides agree on the identities, the negotiated group X25519MLKEM768 and the shared secret.

Both sides of a session end with the same hybrid-derived secret and therefore the same traffic keys, because the transcript binds both key shares and the Finished and the signature are checked over it.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value medium: The document states establishing the same session keys as the first handshake property; the proof holds under an ideal key schedule.
- Stated in the documents: yes. Where the documents say it: rfc9846:6478
- Rests on: the transcript covers both hellos only; HKDF-Extract and HKDF-Expand are ideal one-way functions
- Open: no control lemma: no party in the model derives a different key from the same session
- Next step: Extend the transcript to the full handshake and repeat.

A04: Uniqueness of the handshake session

Each client acceptance corresponds to a different server run, and the same server run cannot be accepted twice.

Each client acceptance is matched injectively by a server run of the same session, so a server flight cannot be replayed into a second acceptance and one server run cannot be accepted twice.

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

- Value medium: The document states uniqueness of the session keys; injectivity of the acceptance is the mechanical form of it.
- Stated in the documents: yes. Where the documents say it: rfc9846:6495
- Rests on: each protocol role instance handles one handshake; fresh randoms on both sides, no record-layer replay window
- Open: no control lemma: a replay would need a party that accepts the same flight twice, which the model does not contain
- Next step: None within this scope.

A06: Key confirmation by the server

At the confirmation point the client requires that the server has derived the same session key and sent a matching Finished.

The client accepts only after the server has shown, with a Finished over the transcript, that it holds the same handshake secret and so the same session keys.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value medium: Key confirmation by the Finished message is stated in the document as a handshake property, and the control shows a client without the check accepts a flight no server produced.
- Stated in the documents: yes. Where the documents say it: rfc9846:6483
- Rests on: HMAC is ideal; the Finished input is the hello transcript only
- Next step: None within this scope.

A08: Transcript binds group and shares

The negotiated group and the server key share that the client accepts must correspond to the group the client proposed, and the transcript binds both roles and the negotiated data.

The key shares and the group the client accepts are those of a real server hello of the same session. A substituted share or a lowered group changes the transcript, so the Finished and the signature no longer verify.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value medium: Downgrade protection and the role of the transcript are stated in the TLS 1.3 security properties; the controls show a client that does not check accepts an attacker-made flight with an attacker-made share.
- Stated in the documents: yes. Where the documents say it: rfc9846:6499, rfc9846:6627
- Rests on: one group is modelled, so the group check is against a fixed name; the transcript covers the hello pair only
- Open: a downgrade to a non-hybrid group needs a second group in the model
- Next step: Add a classical group the client also offers and ask whether a server can force it.

K01: Secrecy of the hybrid session keys

The attacker cannot derive the X25519MLKEM768 shared secret or the handshake and application traffic keys derived from it.

The attacker never derives the hybrid shared secret or the traffic keys of an honest session, and it still does not when it holds one component's secret. It needs both components at once to compute the combined secret.

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

- Value medium: Secrecy of the session keys is the documented property and the component-wise answer is the hybrid property; both are proved with a control in which both component secrets are known.
- Stated in the documents: yes. Where the documents say it: rfc9846:6483, rfc9954:550
- Rests on: ML-KEM is ideal and X25519 has only the law g^xy = g^yx; the combination is one-way in both arguments because HKDF-Extract is a free function of two secrets; the dual-PRF argument is not modelled; no record layer
- Next step: Tamarin+ Agent would be needed for the computational side of the KEM.

C01: Forward secrecy against long-term key leak

After the session completes, leaking the server's long-term signing private key still does not let the attacker obtain the traffic keys of earlier sessions.

The long-term signature key takes no part in the key schedule, so reading it after the handshake gives nothing towards that session's traffic keys.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value medium: Forward secrecy with respect to long-term keys is stated in the TLS 1.3 security properties; the proof is structural and holds with the caveats above.
- Stated in the documents: yes. Where the documents say it: rfc9846:6504
- Rests on: the key schedule is modelled without PSK; the compromise is modelled as the key becoming public after the handshake; erasure of the ephemeral secrets is assumed, the model has no erase step
- Next step: None within this scope.

C04: One ephemeral secret leak keeps keys

When either the X25519 ephemeral private key or the ML-KEM decapsulation private key leaks, the shared secret stays secret.

Either ephemeral secret alone leaves the hybrid shared secret and the session keys secret: the attacker needs both components to compute the combined secret. This is the hybrid property at the symbolic level.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value medium: The hybrid property is stated in the security considerations of both defining documents; the controls show the secret falls once both component secrets are known.
- Stated in the documents: yes. Where the documents say it: rfc9954:550
- Rests on: the combination is a one-way function of the two component secrets, which is where the dual-PRF question sits; a leak is modelled as the scalar or the decapsulation key becoming public at the end of the session; ML-KEM and X25519 are ideal
- Next step: Settle the dual-PRF step with a computational tool.

C08: Hybrid property under component break

When one component key exchange is broken, the shared secret stays secret.

One broken component leaves the hybrid secret safe, and the documented exception holds as well: when one faulty generator supplies both components' secrets, the attacker computes the combined secret and every key derived from it.

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

- Value medium: The hybrid property and its random-number-generator exception are both stated in the main document; the analysis reproduces each with the exact conditions.
- Stated in the documents: yes. Where the documents say it: main:128, main:308
- Rests on: the leak is modelled as one or both component secrets becoming public; the combination is a public function of the two secrets
- Next step: Model generator state rather than the two derived secrets to test the warning more sharply.

K01.10: Secrecy of this handshake's keys

The attacker cannot derive the session keys of this handshake.

The session keys of one handshake are not derivable by the network attacker and stay secret when either component's secret leaks separately; only a single faulty generator feeding both components breaks them.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value medium: The same property as K01, answered for one handshake as the question asked.
- Stated in the documents: yes. Where the documents say it: rfc9846:6483
- Rests on: the same idealisations as the other secrecy answers
- Next step: None within this scope.

A08.11: Finished binds transcript group roles

The Finished that the client accepts binds the full handshake transcript, the negotiated group X25519MLKEM768 and both roles.

The Finished the client accepts is a MAC over the hello transcript under the handshake traffic key, and that transcript carries both key shares and the negotiated group, so group, shares and both roles are bound and a changed group fails the check.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value medium: The document says the Finished input is the transcript hash and that the Finished prevents downgrade; the controls show a client without the check accepts a group and a flight the server never made.
- Stated in the documents: yes. Where the documents say it: rfc9846:6517, rfc9846:6627
- Rests on: the transcript is the hash of the two hellos, not of the whole handshake
- Next step: Add a second group to exercise a real downgrade.

C04.12: One component key leak keeps keys

Leaking the X25519 ephemeral private key or the ML-KEM decapsulation private key separately still leaves the session keys secret.

Leaking the X25519 ephemeral scalar alone, or the ML-KEM decapsulation key alone, leaves the session keys secret. The other component carries the protection.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value medium: The hybrid property the question asks about is stated in the security considerations; the component-wise answers are the ones the model can give.
- Stated in the documents: yes. Where the documents say it: rfc9954:550

- Rests on: ML-KEM and X25519 are ideal and the combination is one-way in both arguments; one leak per session is modelled, published at the end of the session
- Next step: Settle the computational side with Tamarin+ Agent.

X01: Server identity hidden from passive attackers

A passive network attacker observing an honest handshake never learns the server's certificate public key; the certificate and the rest of the server's second flight travel only under the handshake traffic key.

The only trace the prover found has the attacker open a session of its own with the server and receive the certificate as any client would. That says nothing about the passive attacker the goal names, so this analysis neither confirms nor refutes that the certificate stays hidden on the wire.

- Result: No answer; answers what is asked: no; not usable
- Value none: The falsifying trace is an active attacker obtaining public information, so the result is an artifact of the attacker model and says nothing about the protocol.
- Stated in the documents: unclear. Source text it rests on: rfc9846:6530, s0022
- Rests on: the model has one server identity and an active attacker, so the public certificate is always obtainable by connecting as a client
- Open: whether a passive observer sees the server's certificate
- Next step: Build a passive-only attacker over a model with two candidate server identities and ask the non-interference question.

X02: Hybrid secret survives a shared RNG leak

When the state or output of the random number generator used for a session is disclosed to the attacker and both component algorithms draw from that same generator, the hybrid shared secret and the session keys derived from it stay secret.

When the same faulty generator supplies both components' secrets, the attacker computes the combined secret and every key derived from it. This is the warning of the main document's security considerations, reproduced in the model.

- Result: Attack holds (stated in the documents); answers what is asked: yes; usable with caveats
- Value low: The failure follows directly from the assumed compromise, and the document states it as a warning to implementers.
- Stated in the documents: yes. Where the documents say it: main:308
- Rests on: the leak is modelled as both component secrets becoming public; the combination is a public function of the two secrets
- Next step: Model the generator itself to show how far the warning reaches.

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_ident	4 of 6		6 proved
A03	proved		a03_key			1 proved
A04	proved		a04_inj			1 proved
A06	proved		a06_fin	checked		1 proved
A08	proved		a08_share	checked		2 proved
K01	proved		k01_hs	checked		5 proved
C01	proved		k01_hs	checked		1 proved
C04	proved		c04_x	checked		2 proved
C08	proved		c04_x	checked	c08_both (holds_in_source)	2 proved, 1 attack
K01.10	proved		k01_hs	checked		3 proved
A08.11	proved		a06_fin	checked		2 proved
C04.12	proved		c04_x	checked		2 proved
X01	none				ident (model_artifact)	1 attack
X02	none				c08_both (holds_in_source)	1 attack

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

target	claim	attacker	answer	query	control
A02	The peer identity in the client's acceptance is the identity the client is configured to trust: any certificate public key the client accepts equals the server's own public key, so the client's view of the peer reflects the server's identity (the target's "intended identity").	`network`	proved	a02_ident	none
A02	Every acceptance by the client is backed by a certificate that the server really sent in that same session, so a server run of the session exists.	`network`	proved	a02_cert	none
A02	The client accepts only the pinned server identity, although the attacker also holds a valid signing key and certificate of a different server identity and can run a complete server flight with it.	`rogue_server_key`	proved	a02_ident	checked
A02	The certificate the client accepted came from the honest server in the same session, not from the attacker's own server identity, although the attacker holds another server's signing key.	`rogue_server_key`	proved	a02_cert	checked
A02	The peer identity in the client's acceptance is the identity the client is configured to trust: any certificate public key the client accepts equals the server's own public key.	`network`	proved	a02_ident	checked

target	claim	attacker	answer	query	control
A02	Every acceptance by the client is backed by a certificate that the server really sent in that same session.	`network`	proved	a02_cert	checked
A03	When a client session and a server session of the same handshake both complete, they hold the same hybrid-derived handshake secret, hence the same session keys and the same negotiated X25519MLKEM768 shares.	`network`	proved	a03_key	none
A04	Each client acceptance is matched injectively by a server role run of the same session that sent its Finished over the same keys, so one server run cannot be accepted twice and no old flight can be accepted again.	`network`	proved	a04_inj	none
A06	The client accepts only if a server run of the same session derived the same hybrid handshake secret and sent a Finished over the transcript with it: key confirmation by the server.	`network`	proved	a06_fin	checked
A08	The X25519 and ML-KEM key shares the client accepts are the ones of a real server hello of the same session; a substituted share changes the transcript and the handshake secret, so the Finished and the signature fail.	`network`	proved	a08_share	checked
A08	The negotiated group the client accepts is the X25519MLKEM768 group the server named in the same session, not one of the attacker's choosing.	`network`	proved	a08_grp	checked
K01	The attacker does not learn the hybrid-derived handshake secret (hence the session keys) of an honest session that leaks no ephemeral secret.	`network`	proved	k01_hs	checked
K01	The attacker does not learn the client application traffic key derived from the hybrid shared secret of an honest session that leaks no ephemeral secret.	`network`	proved	k01_ap	checked
K01	The attacker knows the client's X25519 ephemeral scalar (hence that component's shared secret) and still does not learn the hybrid shared secret or the keys derived from it.	`x25519_ephemeral_leak`	proved	c04_x	checked
K01	The attacker knows the client's ML-KEM decapsulation key (hence the ML-KEM shared secret) and still does not learn the hybrid shared secret or the keys derived from it.	`mlkem_leak`	proved	c04_kem	checked
K01	The attacker reads the server's long-term signature private key after the handshake and still does not learn the session keys of the completed session.	`server_ltk_after`	proved	k01_hs	checked
C01	Obtaining the server's long-term signature private key after the handshake completes does not give the traffic keys of that earlier session: the key schedule takes no long-term key as input.	`server_ltk_after`	proved	k01_hs	checked

target	claim	attacker	answer	query	control
C04	With the client's X25519 ephemeral private scalar leaked, the hybrid shared secret and the session keys stay secret because the ML-KEM component still protects them.	`x25519_ephemeral_leak`	proved	c04_x	checked
C04	With the client's ML-KEM decapsulation private key leaked, the hybrid shared secret and the session keys stay secret because the X25519 component still protects them.	`mlkem_leak`	proved	c04_kem	checked
C08	The X25519 component key exchange is broken (its shared secret is known to the attacker) and the hybrid shared secret stays secret.	`x25519_ephemeral_leak`	proved	c04_x	checked
C08	The ML-KEM component key exchange is broken (its shared secret is known to the attacker) and the hybrid shared secret stays secret.	`mlkem_leak`	proved	c04_kem	checked
C08	The hybrid shared secret stays secret when the random-number generator state used for the session is disclosed and both component algorithms draw from that same generator.	`rng_shared`	attack (holds_in_source)	c08_both (the control of c04_kem)	
K01.10	The attacker does not derive the session keys of this handshake (X25519MLKEM768 shared secret and the traffic keys derived from it).	`network`	proved	k01_hs	checked
K01.10	The session keys stay secret when the X25519 ephemeral scalar is leaked separately.	`x25519_ephemeral_leak`	proved	c04_x	checked
K01.10	The session keys stay secret when the ML-KEM decapsulation key is leaked separately.	`mlkem_leak`	proved	c04_kem	checked
A08.11	The client's accepted Finished is computed over the handshake transcript with the handshake traffic key, so it binds both roles and the negotiated data of this session.	`network`	proved	a06_fin	checked
A08.11	The accepted Finished is bound to the X25519MLKEM768 group the server named in this session, so the group cannot be lowered without the Finished failing.	`network`	proved	a08_grp	checked
C04.12	The session keys stay secret when only the X25519 ephemeral scalar leaks.	`x25519_ephemeral_leak`	proved	c04_x	checked
C04.12	The session keys stay secret when only the ML-KEM decapsulation key leaks.	`mlkem_leak`	proved	c04_kem	checked
X01	A passive network attacker observing an honest handshake never learns the server's certificate public key.	`network`	attack (model_artifact)	ident	
X02	The hybrid shared secret and the session keys stay secret when the RNG state shared by both components is disclosed to the attacker.	`rng_shared`	attack (holds_in_source)	c08_both (the control of c04_kem)	

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-20261114T182015, 2026-11-14. Status: finished. Model: deepseek-v4-1-flash-261017. Steps: 40, prover 62 s, wall 18 min, cost 0.130 USD.

ProVerif: 82 proofs run, 81 decided, 1 without an answer, 42 s, 38 answers recorded.

What was analysed

Protocol	TLS 1.3: X25519MLKEM768 hybrid key exchange
Version	rfc10024
Analysed	Fixed X25519MLKEM768; first a single full 1-RTT server-authenticated handshake is selected, the authentication configuration is fixed, and PSK/0-RTT/resumption/HRR are added separately afterwards.
Not analysed	The extension defines only key_share and the shared secret; the full handshake must be read from the TLS baseline; the symbolic KEM/DH results cannot serve as a proof of the computational security or quantum resistance of the ML-KEM algorithm. Only the properties under the explicit model and attacker conditions are proved; missing checks, a complete secure channel or an all-honest NF cannot substitute for the conclusions to be proved.
Source text	621 numbered passages (164414 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
rfc9954	6	Security Considerations
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

Document	Section	Title
rfc9846	4.2	Key Exchange Messages
rfc9846	4.2.1	Cryptographic Negotiation
rfc9846	4.2.2	Client Hello
rfc9846	4.2.3	Server Hello
rfc9846	4.2.4	Hello Retry Request
rfc9846	4.3	Extensions
rfc9846	4.3.1	Supported Versions
rfc9846	4.3.2	Cookie
rfc9846	4.3.3	Signature Algorithms
rfc9846	4.3.4	Certificate Authorities
rfc9846	4.3.5	OID Filters
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

Document	Section	Title
rfc9846	4.7.3	Key and Initialization Vector Update
rfc9846	7	Cryptographic Computations
rfc9846	7.1	Key Schedule
rfc9846	7.2	Updating Traffic Secrets
rfc9846	7.3	Traffic Key Calculation
rfc9846	7.4	Asymmetric Shared Secret Calculation
rfc9846	7.4.1	Finite Field Diffie-Hellman
rfc9846	7.4.2	Elliptic Curve Diffie-Hellman
rfc9846	7.5	Exporters

The protocol's purpose, as the source step located it in the documents: fix X25519MLKEM768; first select one full 1-RTT server-authenticated handshake, fix the authentication configuration, and add PSK/0-RTT/resumption/HRR later, each separately.

The properties against the documents

Before writing a query, 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 for the client	the client's assurance, when it accepts, that the peer it authenticated is the server whose signature key it pins, and that a server run of this session exists	rfc9846 Appendix F.1: peer authentication means the client's view of the peer identity reflects the server's identity; the server's signature over the handshake transcript and the Finished MAC tie the session to the server's identity.	rfc9846:6490, rfc9846:6483, s0065, s0083, s0092
A03	Session key agreement between peers	the client and the server of the same session ending with the same session keys and the same negotiated group X25519MLKEM768	rfc9846 Appendix F.1: establishing the same session keys, the handshake must output the same session keys on both sides; the hybrid shared secret is the concatenation of the ML-KEM and X25519 shared secrets inserted into the TLS 1.3 key schedule.	rfc9846:6478, s0015, s0052, s0092
A04	Uniqueness of the handshake session	each client acceptance belongs to a distinct server run, and one server run cannot be accepted twice	rfc9846 Appendix F.1: any two distinct handshakes should produce distinct, unrelated session keys; the Finished MAC and the fresh randoms of both hellos prevent a replayed flight from being accepted again.	rfc9846:6495, s0092, s0507, s0564
A06	Key confirmation by the server	the client's acceptance requires the server to have derived the same session keys and to have sent a matching Finished	rfc9846 Appendix F.1 and Section 4.5.3: the Finished message provides key confirmation for the shared secrets established in the handshake and a recipient MUST verify its contents.	rfc9846:6483, rfc9846:6507, s0092, s0507, s0513

ID	Name	About	What the documents say	Where
A08	Transcript binds group and shares	the group and the server key share the client accepts are the ones of the negotiated X25519MLKEM768 run, bound to both roles by the handshake transcript	rfc9846 Appendix F.1: downgrade protection, the cryptographic parameters should be the same on both sides and as if no attack had happened; the Finished MAC and the signature prevent downgrade attacks. main Section 6: the analysis relies crucially on the TLS 1.3 message transcript.	rfc9846:6499, rfc9846:6627, main:268, s0003, s0008, s0015, s0092
K01	Secrecy of the hybrid session keys	the X25519MLKEM768 shared secret and the handshake and application traffic keys derived from it, kept from the network attacker	rfc9846 Appendix F.1: secrecy of the session keys, they should be known only to the communicating parties; the hybrid property requires the secret to stay secure as long as at least one component key exchange is unbroken.	rfc9846:6483, rfc9954:550, s0015, s0052, s0053, s0065
C01	Forward secrecy against long-term key leak	the session keys of a completed handshake, kept from an attacker that obtains the server's long-term signature private key afterwards	rfc9846 Appendix F.1: forward secret with respect to long-term keys, compromise of the signature keys after the handshake is complete does not compromise the security of the session key. The TLS 1.3 key schedule takes no long-term key as input.	rfc9846:6504, s0583
C04	One ephemeral secret leak keeps keys	the hybrid shared secret and the session keys, kept when either the client's X25519 ephemeral scalar or its ML-KEM decapsulation key leaks	rfc9954 Section 6: the hybrid property, the resulting secret is secure as long as at least one of the component key exchange algorithms is unbroken; main Section 6 discusses the disclosure of the classical scalar and of the ML-KEM encapsulation randomness.	rfc9954:550, main:299, s0025, s0057
C08	Hybrid property under component break	the hybrid shared secret of one session, kept when one component key exchange (X25519 or ML-KEM) is broken and the other is not	main Section 3 defines hybrid as security as long as at least one component algorithm remains secure, even if the others are compromised; rfc9954 Section 6 states the same requirement and names the dual-PRF combiner.	main:128, rfc9954:550, s0020, s0057
K01.10	Secrecy of this handshake's keys	the session keys of the one handshake analysed, kept from the network attacker and from an attacker that obtains one ephemeral secret	rfc9846 Appendix F.1 secrecy of the session keys; the user question Q1 asks for the session key of this handshake.	rfc9846:6483, s0015, s0052, s0053
A08.11	Finished binds transcript group roles	the client's accepted Finished covering the handshake transcript, the negotiated X25519MLKEM768 group and both roles	rfc9846 Section 4.5.3: verify_data is an HMAC over the transcript hash of the handshake context; Appendix F.1: the Finished MAC prevents downgrade attacks. User question Q2.	rfc9846:6517, rfc9846:6627, s0092, s0507
C04.12	One component key leak keeps keys	the session keys, kept when either the X25519 ephemeral scalar or the ML-KEM decapsulation key leaks separately	rfc9954 Section 6 hybrid property; main Section 6 on the two component secrets. User question Q3.	rfc9954:550, main:299, s0023, s0025, s0057
X01	Server identity hidden from passive attackers	the server's identity (its certificate public key) and the passive network attacker	rfc9846 Appendix F.1: the server's identity (certificate) should be protected against passive attackers; the client's identity against passive and active attackers.	rfc9846:6530

ID	Name	About	What the documents say	Where
X02	Hybrid secret survives a shared RNG leak	the X25519MLKEM768 shared secret of one session against an attacker that obtains the output of the RNG shared by both components	main Section 6 warns that if the same insecure RNG is used by both algorithms, a disclosure of state by one of the algorithms will also affect the security of the other; the encapsulation randomness is an additional place where RNG output is disclosed. The property the warning says is lost; expected to fail.	main:308, main:282, s0023, s0026

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

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: 10. Not taken: 10. 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, A03	
P02	main:273-276	The prover cannot be asked	Guidance to implementers on side-channel resistant KEM implementations, with a pointer to NIST SP 800-227; side channels and implementation quality are outside what a symbolic trace prover can be asked.
P03	main:278-280	A03	
P04 to P05	main:282-297	`rng_shared`	
P06	main:299-306	K01, C04, `x25519_ephemeral_leak`	
P07	main:308-310	X02, `rng_shared`	
P08	rfc7748:783-793	The prover cannot be asked	The brute-force security level of curve25519 as a primitive; the strength of a primitive is not a trace property.
P09	rfc7748:802-808	The prover cannot be asked	A performance/paranoia trade-off between curve25519 and curve448 and the strength of both against large quantum computers; not a trace property.
P10	rfc7748:810-818	The prover cannot be asked	Contributory behaviour of X25519: a small-order input point eliminates the other party's contribution and is detected by the all-zero shared secret check. A model whose only DH law is g^xy = g^yx cannot represent small-order points, so this validity check is left out of the model and is listed in the limits.
P11	rfc7748:820-825	A08	
P12	rfc7748:827-837	The prover cannot be asked	Implementations that reject points on the twist or non-minimal field elements may be trivially distinguishable from the Montgomery ladder; this is observable implementation behaviour, not a trace property of the protocol.
P13	rfc9846:4676-4677	No goal stated	A pointer saying that security issues are discussed in Appendix C, Appendix E and Appendix F of rfc9846; it states no property of its own.
P14	rfc9954:550-558	C08, C04, K01	

Paragraphs	Where	Taken up as	Why not
P15	rfc9954:560-569	The prover cannot be asked	The requirement that a KEM used here be IND-CCA2 secure or carry a Fujisaki-Okamoto transform, and that some IND-CPA KEMs are broken under public key reuse: this is computational security of the KEM, which the symbolic model assumes ideal.
P16	rfc9954:571-574	A03	
P17	rfc9954:576-582	The prover cannot be asked	Timing side channels of variable-length secrets processed by a hash function. The model has no notion of length or of time.
P18	rfc9954:584-586	The prover cannot be asked	Timing attacks (Lucky Thirteen, Raccoon) on variable-length secrets; not a trace property.
P19	rfc9954:588-592	The prover cannot be asked	A risk of variable-length secrets when the KDF hash is not collision resistant; a computational assumption about the hash, not a trace property.
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: 11. Left out: 22. 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	Outside the scope	Payload and end-to-end confidentiality of application data belong to the record layer; this task fixes the extension's key exchange and one full 1-RTT handshake.
K03	Strong secrecy / indistinguishability of secret values	K01	
K04	Key purpose, direction and session separation	A03, A04	
D01	Message integrity and data origin authenticity	A03, A08	
A01	Peer aliveness	A02	
A05	Implicit key authentication	A03, K01	
A07	Identity–credential–key binding	A02	
R01	Authentication / challenge freshness	A04	
R02	Message-level anti-replay	A04	
R03	At most once per business operation	Not applicable	The handshake carries no billing, token or allocation operation that could be repeated.
R04	Message acceptance order / window security	Outside the scope	Acceptance order and anti-replay windows are record-layer behaviour; this task models the handshake messages only.
R05	Anti-downgrade / security policy preservation	A08	

Entry	Property of the catalogue	Examined as	Why not
C02	Resistance to key compromise impersonation (KCI)	Outside the scope	Key compromise impersonation is about the compromised actor's authentication of its peer; this configuration is server-authentication only, with no client long-term key, so the client-key direction does not exist and the server-key direction is the ordinary ability of that server.
C03	Known-key security / leak isolation	A04, K01	
C05	Post-compromise security (PCS)	Outside the scope	Post-compromise security needs a ratchet or resumption; PSK, 0-RTT, resumption and key update are explicitly deferred by the task's procedure.
C06	Augmented authentication after compromise of enrolment credentials	Not applicable	There is no registration record, PAKE or augmented authentication in TLS 1.3 certificate-based key exchange.
C07	Resistance to offline dictionary attacks	Not applicable	No password or other low-entropy shared secret is used in this configuration.
Z01	Correct enforcement of authorization	Not applicable	The protocol carries no authorization decision or protected resource access.
Z02	No expansion of delegated permissions	Not applicable	No delegation or proxy of permissions exists in the handshake.
Z03	Binding of authorization to identity / session / operation	Not applicable	No authorization token is issued or moved between contexts.
Z04	Revocation and expiry take effect	Not applicable	No revocable permission, expiry or quota is part of the exchange.
Z05	Cross-tenant / cross-service security isolation	Not applicable	There is no multi-tenant or cross-service authorization boundary in the handshake.
S01	Encryption nonce uniqueness	Outside the scope	AEAD nonce uniqueness is a record-layer obligation; the handshake model derives keys but sends no record-layer traffic.
S02	Retransmission / key reinstallation security	Outside the scope	Key reinstallation and retransmission belong to the record layer and to KeyUpdate, and KeyUpdate is deferred by the task's procedure; the handshake itself is one-shot and holds no installed key that a retransmission could reset.
S03	Resistance to rollback and state monotonicity	Outside the scope	There is no cached version, counter or resumption state in this configuration that an old message could roll back.
S04	Cross-phase security context consistency	Outside the scope	Cross-phase context consistency needs resumption, 0-RTT or migration, all deferred by the task's procedure.
S05	Termination and invalidation of old state	Outside the scope	Connection teardown and invalidation of old state belong to the record layer and are deferred with KeyUpdate.
P01	Identity anonymity	X01	
P02	Session unlinkability	Outside the scope	Session unlinkability is not a goal of TLS 1.3 and is not in the task's scope; the server is authenticated by a certificate, so two sessions with the same server are recognisable by design.
P03	Hiding of user / credential existence	Not applicable	No user database or credential lookup takes place in the handshake.
V01	Conditional progress / error recovery	The prover cannot be asked	Conditional progress and error recovery are liveness properties; a trace prover cannot be asked whether a run always ends.
V02	No unexpected deadlock	The prover cannot be asked	Deadlock freedom is a liveness property outside what ProVerif can be asked.

Entry	Property of the catalogue	Examined as	Why not
V03	Bounded resource abuse / resistance to amplification	The prover cannot be asked	Resource amplification and cost bounds are quantitative properties outside what ProVerif can be asked.

The parties and the protected values, one by one

The host listed the 16 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: 4. Without: 12. The list follows from the properties; what was made of each entry is the analyst's judgement.

ID	Party or value	Kind	Attacker	Why none
S01	Client	party	`x25519_ephemeral_leak`, `mlkem_leak`, `rng_shared`	
S02	Server	party	`server_ltk_after`	
S03	Server identity	value	Nothing gained	The server's certificate is public information that any party can obtain out of band. Holding it gives nothing towards the secrecy or agreement targets. Whether it appears on the wire is the separate goal added from Appendix F.1, which is asked under a passive network attacker rather than under an attacker that already holds the value.
S04	Negotiated group	value	Nothing gained	The negotiated group is carried in the clear in both hellos by design.
S05	Shared secret	value	Everything lost by definition	A party that holds a session's hybrid shared secret is by definition a party to that session. The secrecy targets that name it are lost by definition, and the documents ask no authentication property against a party holding the session's own keys.
S06	Both identities	value	Nothing gained	The server's identity is public and the client's identity is not authenticated in this configuration, so holding either gives nothing.
S07	Session uniqueness	value	Nothing gained	Session uniqueness is a property of the run, not a value an attacker can hold. The way to attack it is a replay, which is asked as A04.
S08	Handshake traffic key	value	Everything lost by definition	Holding a session's handshake traffic key is exactly what K01 and C04 deny. Once it is held those properties are lost by definition and no document asks anything further under it.
S09	Key share	value	Nothing gained	Both key shares travel in the clear in the hellos by design; the attacker on the network already has them.
S10	Transcript	value	Nothing gained	The transcript of both hellos is public by design; the attacker on the network already has it, and binding to it is what A08 asks.
S11	ML-KEM shared secret	value	`mlkem_leak`	
S12	X25519 shared secret	value	`x25519_ephemeral_leak`	
S13	Application traffic key	value	Everything lost by definition	The application traffic key is the value whose secrecy K01 and C01 state. Holding it loses those by definition, and application-data protection is outside this scope.
S14	Historical session traffic keys	value	Everything lost by definition	Historical session traffic keys are exactly what C01 says must not follow from the long-term key; an attacker that already holds them loses that property by definition.

ID	Party or value	Kind	Attacker	Why none
S15	Session keys	value	Everything lost by definition	The session keys are the object of K01 and K01.10; holding them loses those targets by definition.
S16	Role	value	Nothing gained	Roles are bound into the transcript by the Finished and the signature. There is no separate role value an attacker can hold, and the binding is asked as A08.

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		
`x25519_ephemeral_leak`	the client's X25519 ephemeral private scalar of the session, and hence that session's X25519 shared secret; obtained during or after the handshake	The hybrid property must hold if one component key exchange is broken (rfc9954 Section 6), and main Section 6 says a passive observer with a cryptographically relevant quantum computer can recover the X25519 scalar, which is derived from RNG output. This is the broken-classical-component attacker.	rfc9954:550, main:299, s0025, s0057
`mlkem_leak`	the client's ML-KEM-768 decapsulation private key of the session, and hence that session's ML-KEM shared secret	The hybrid property must hold if one component key exchange is broken (rfc9954 Section 6); the target C04 names the ML-KEM decapsulation key as one of the two secrets whose separate leak must not break the session. This is the broken-post-quantum-component attacker.	rfc9954:550, main:128, s0057, s0058
`rng_shared`	the state or output of the random number generator shared by both component algorithms for the session, hence the X25519 ephemeral scalar and the ML-KEM encapsulation randomness of that session	main Section 6: the encapsulation randomness m is drawn from a random bit generator and is recovered by the client, so it is RNG output disclosed to a party; if the same insecure RNG is used by both algorithms, a disclosure of state by one also affects the other.	main:308, main:282, main:289, s0023, s0024, s0026
`server_ltk_after`	the server's long-term signature private key, read from where the server keeps it, during a session or after the handshake has completed	rfc9846 Appendix F.1 asks forward secrecy with respect to long-term keys: compromise of the signature keys after the handshake is complete must not compromise the session key. The long-term signature key is the only long-term key in this configuration.	rfc9846:6504, s0583
`rogue_server_key`	a valid signature private key and certificate for a server identity other than the one the client is configured to accept	Peer authentication (rfc9846 Appendix F.1) is a statement about the client's view of the server's identity, so the attacker must be able to offer a certificate of another identity; any server holds one of its own.	rfc9846:6490

ID	`network`	`x25519_ephemeral_leak`	`mlkem_leak`	`rng_shared`	`server_ltk_after`	`rogue_server_key`
A02	Holds					Holds

ID	network	x25519_ephemeral_leak	mlkem_leak	rng_shared	server_ltk_after	rogue_server_key
A03	Holds					
A04	Holds					
A06	Holds					
A08	Holds					
K01	Holds	Holds	Holds		Holds	
C01					Holds	
C04		Holds	Holds			
C08		Holds	Holds	Fails		
K01.10	Holds	Holds	Holds			
A08.11	Holds					
C04.12		Holds	Holds			
X01	No answer					
X02				Fails		

Source and targets

Source: 621 units, 164414 bytes, frozen in source-delivery-01.json. Targets: 9 approved (A02, A03, A04, A06, A08, K01, C01, C04, C08); 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 the source package protocol-62b1d6c019165ab627bd9496162f

This step as it reports itself: 1 requests (1 failed), 0.000 USD, 74179 tokens in and 3987 out, 24 s.

Notes on the prover and the model

What the analyst named for targets:

step	targets	features	reason
29	A02, A03, A04, A06, A08, K01, C01, C04, C08, K01.10, A08.11, C04.12, X01, X02	plain_primitives	Every answer rests on idealised primitives: ML-KEM as a perfect encapsulation (decap(encaps(ek,s))=s, no computational or quantum claim), X25519 as Diffie-Hellman with only the law g^xy = g^yx (no contributory behaviour, no small-order points), HKDF-Extract/Expand as one-way functions, HMAC and AEAD as ideal. The hybrid property holds in the model because HKDF-Extract is a public function of both component secrets, so the attacker needs both; the dual-PRF argument that real security needs (rfc9954 Section 6) is not modelled, and no record layer is present. Tamarin+ Agent, which models these features as they are, would be needed to settle the computational side.

The analyst's closing note

Modelled the TLS 1.3 full 1-RTT server-authenticated handshake with the X25519MLKEM768 hybrid group in ProVerif: ClientHello(key_share = ML-KEM ek || X25519 share) / ServerHello(key_share = ct || X25519 share) / encrypted EncryptedExtensions, Certificate, CertificateVerify, Finished / client Finished, with the hybrid secret as HKDF-Extract over the two component secrets and the hello transcript as context. The plain run was found first; the model then grew by one mechanism at a time (encrypted server flight, leak families for the X25519 scalar, the ML-

KEM decapsulation key and a single faulty shared RNG, a rogue server key, a post-handshake reveal of the long-term signature key, and an unchecked client used only to control the binding properties). No target rests on a value that changes across steps, so no bounded model and no state review were needed.

Results: 26 answers recorded over 14 targets, all still holding on the final text. Twelve targets are proved: server authentication and the pinned identity (with a control where an unchecked client accepts the attacker's own certificate), agreement on the same session keys, injectivity of the acceptance, key confirmation by the Finished, binding of the key shares and the group to the real server hello, secrecy of the hybrid secret and of the traffic keys, forward secrecy against the long-term signature key, and — the central result — the hybrid property: leaking either component's secret alone leaves the shared secret and the session keys secret, proved by a model in which HKDF-Extract needs both components, with a control in which both are known. Two results are attacks that the documents state themselves: one shared faulty generator breaks the hybrid property (main Section 6), recorded as a known result. One target, identity protection against a passive attacker, stays open: the only trace ProVerif found has an active attacker obtaining the public certificate as any client would, which is an artifact of an active-attacker model with a single server identity.

What blocks the rest: the symbolic treatment of ML-KEM and X25519 (single DH law, no small-order or all-zero check), the hello-only transcript, and the absence of a passive-only attacker. The computational side of the hybrid combiner (the dual-PRF assumption) needs Tamarin+ Agent; the answers are reported as resting on those simplifications.