Security analysis of TLS hybrid PSK-DHE resumption profile

rfc10024 · ProVerif+ Agent report

Documents	Post-Quantum Traditional (PQ/T) Hybrid Key Agreement Mechanisms for TLS 1.3 (protocol definition); The Transport Layer Security (TLS) Protocol Version 1.3 (supporting)
Properties examined	12: 12 hold, 0 fail
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 profile joins an X25519MLKEM768 key exchange with TLS 1.3 resumption. A client and a server first run a full handshake authenticated by the server's certificate; the server then hands out a ticket, and the next connection is resumed with psk_dhe_ke using the PSK that ticket stands for. The client holds the pinned server certificate key, its ephemeral X25519 scalars and its ML-KEM key pairs; the server holds its certificate and signature key and the tickets it issues.

When each exchange ends, both sides hold the same handshake and application traffic keys, know the same ticket identity and have confirmed that the peer computed the same transcript. On the resumption the client is sure it is talking to the server whose certificate it checked in the original handshake, and the server is sure the client held the ticket and the binder, although neither certificate nor signature is sent again.

The parties:

- **Client**: X25519MLKEM768 client, resumption PSK holder. Holds: server certificate (pinned public key).
- **Server**: X25519MLKEM768 server, ticket issuer. Holds: certificate and signature key sks, ticket store.

Figure 1 shows the exchange as the documents specify it.

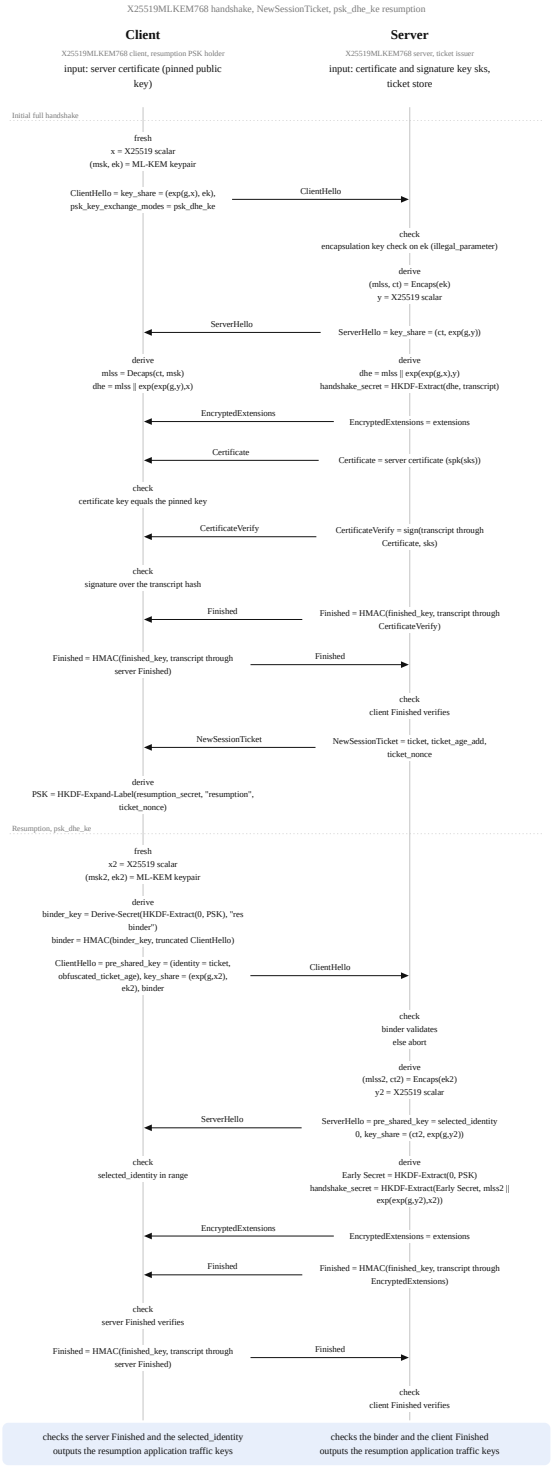


Figure 1. The protocol as specified.

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

2 Findings

No finding. The analysis found nothing that the documents do not already state or set aside. The table in section 3 has the result for every property.

Within the scope analysed the profile keeps every property the targets ask for: all twelve targets are proved, none is falsified, and there is no finding. What the documents state about agreement, secrecy, forward secrecy with respect to the resumption PSK, the hybrid property under a single leaked key exchange component, and ticket and binder binding is reproduced mechanically on one full handshake followed by one resumption. The results rest on the simplifications of the model: one client and one server run, a deterministic KEM, one-way hash functions the attacker cannot apply, and no replay store. A prover that models a server keeping tickets across sessions is needed before anything can be said about repeated use of one ticket.

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
A03	**resumption session key agreement.** Whenever the client (or server) is presentpsk_dhe_keBack in hand to receive the conversation, there's a match between the two sides on each other's identity and choice.PSKStatus, Consultative Group andkey_shareAnd the handshake key from Transcript is identical.	no compromise	Holds	medium	yes
A04	**no replay of a resumption flight.** Each successful resumption of a handshake is matched by a different cross-end; The same ClientHello/binder or the same Finished cannot be accepted twice.	no compromise	Holds	low	yes
A08	**resumption parameters not substitutable.** Reinstatement of consultative parameters accepted in handshakes (groups, groups)key_share、PSKIdentity, Transcript) proposal for and context of this session cannot be replaced by other sessions or otherPSKvalue.	no compromise	Holds	medium	yes
K01	**secrecy ofPSKand traffic keys.** The assailant is not able to extrapolatePSK、binder_key, or apply the flow key.	no compromise	Holds	medium	yes
C01	**PSKleak spares earlier session.** Visible after resumption of sessionPSK, does not expose previously completed session keys or applied data.	Resumption PSK (after completion of session)	Holds	medium	yes
C03	**resumption key isolated fromPSKand original key.** DisclosurePSKor the original session applies the flow key and does not disclose the application flow key for the resumed session.	Resumption PSK or application of flow keys for original sessions (restored during or after handshake)	Holds	medium	yes
C04	**one brokenKEXcomponent spares keys.** Recovered by leakX25519Temporary private key orML-KEMOne of the private keys, resume the session key remains confidential.	X25519Temporary private key orML-KEMPrivate key (recovering handshake period)	Holds	medium	yes
S04	**resumption inherits the original context.** The identity of the resumed session, PSK and the context of the key correctly inherited from the original handshake and was not misdirected to other sessions.	no compromise	Holds	low	yes
A08.9	**PSKand binder tied to server and transcript.** Recovered accepted in handPSK/binder binds to the original server identity and restores transcript.	no compromise	Holds	medium	yes

ID	Property	Compromise assumed	Result	Value	In the documents
U01	**single leak leaves resumption key secret.** Separate DisclosurePSKOr a new one.KEXScale, whether the restoration key remains confidential.	resumptionPSKOr a new one.KEXScale (restoration of handshake period)	Holds	medium	yes
C03.11	**application keys isolated between connections.** Restoring the session application key is isolated from the original session application key and the leaking party does not disclose the other party.	Original session or resumed session application flow key (recovering handshake period or later)	Holds	medium	yes
X01	**distinct handshakes give distinct session keys.** No two distinct sessions derive the same resumption application traffic key: if the client's session and the server's session name the same key, the two sessions are the same session.	no compromise	Holds	low	yes

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

The questions asked, and the analyst's answers:

Question	Answer	Rests on
Q1: Restore PSK/ binder bound to original server identity and restore transcript	Yes: the binder the server accepts binds exactly the ClientHello the client sent, and the client only takes a PSK from a session in which it verified the server's certificate key and the CertificateVerify signature, so the PSK and binder are tied to the original server identity and to the resumption transcript.	A08.9, S04, A08
Q2:PSKOr a new one.KEXIndependent effects of mass leakage	Leaking the resumption PSK alone, the X25519 ephemeral scalar alone, or the ML-KEM decapsulation key alone never gives the resumption session keys, because the handshake secret is extracted from the PSK, the concatenation of both key exchange results and the transcript.	U01, C04, C01
Q3: Restore isolation from original session application keys	The two connections derive their application traffic keys from different master secrets through one-way derivations, so holding the original connection's application traffic key does not give the resumption connection's key, and holding the resumption key does not give the original server-to-client key; the remaining direction is not answered.	C03.11, C03, C01

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:

- One client process and one server process, no replication: replay of a ClientHello, a binder or a Finished into a second session, and the reuse of one ticket for several resumptions, are outside the model.
- The KEM is deterministic (encapsulation takes no randomness) and its decapsulation key is a plain name, so KEM fault attacks, ciphertext-length checks and the encapsulation-key check are not modelled beyond aborting.
- Transcript hashes, concatenation and the hash used by the KDF are private functions the attacker cannot apply himself; the attacker is therefore slightly weaker than reality on values he knows.
- X25519 is modelled by the single law g^xy = g^yx; no other algebraic property of the curve, and no all-zero shared secret check, is represented.
- The ticket age is carried but not checked: ticket_lifetime, the de-obfuscation of obfuscated_ticket_age and any acceptance window are not modelled, and no target rests on them.
- 0-RTT, cross-cluster ticket distribution, certificate-based client authentication, key updates and post-handshake messages are outside the scope of the task and not modelled.

- The leaks of PSK, X25519 scalar, ML-KEM decapsulation key and traffic keys are placed in separate phases after the run, and each leak query also includes the earlier leaks of the ladder; an attacker who obtains a value in the middle of a session and interferes with it is only approximated.
- The direction in which the resumption traffic key leaks and the original connection's client-to-server key must stay secret is not answered; only the server-to-client key of the original connection is covered there, because one model cannot leak a value and keep it secret at the same time.

What the host records:

- The answers changed after the reading was written for: A04, K01, C03, U01.

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.

A03: resumption session key agreement

Every time a client (or server) is inpsk_dhe_keBack in hand to receive the conversation, there's a match between the two sides on each other's identity and choice.PSKStatus, Consultative Group andkey_shareAnd the handshake key from Transcript is identical.

In a resumption the two sides end with the same handshake key, and that key is derived from the PSK, both key shares and the whole transcript, so they agree on the selected PSK identity, the shares and the transcript-derived keys. The server's acceptance implies a client ran the matching exchange, the client's acceptance implies the server sent the Finished under the same key, and the control shows the query is able to fail when the client's message never reaches the server.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value medium: RFC 9846 Appendix F.1 asks that a completed handshake output the same session keys on both sides; the model confirms it for the resumption and adds no weakness.
- Stated in the documents: yes. Where the documents say it: rfc9846:6478
- Rests on: one client and one server process, no repetition of the run; X25519 by g^xy = g^yx only; the transcript is a private hash chain the attacker cannot recompute
- Next step: a model with several concurrent sessions would test the same agreement across sessions

A04: no replay of a resumption flight

Each successful resumption of handshake is matched by a different cross-end; The same ClientHello/binder or the same Finished cannot be accepted twice.

The answers of this target changed after this reading was written.

Each server acceptance of a client Finished corresponds to a distinct client run, so one Finished is not accepted twice within the modelled exchange. The model has a single server session and no ticket store, so it says nothing about a client replaying one ClientHello and binder, or one ticket, into a second session.

- Result: Holds; answers what is asked: partly; usable with caveats
- Value low: RFC 9846 Appendix F.1 asks for unique session keys between distinct handshakes; the injective agreement here follows from fresh key shares per session rather than from a replay check.
- Stated in the documents: yes. Where the documents say it: rfc9846:6495
- Rests on: one server process, no replication; no ticket store and no single-use enforcement modelled, as the task puts replay storage out of scope
- Open: acceptance of the same ticket and binder in two server sessions
- Next step: unroll a server that keeps a ticket store over several resumptions

A08: resumption parameters not substitutable

Restoring consultative parameters accepted by handshakes (groups, groups)key_share、 PSKIdentity, Transcript) proposal for and context of this session cannot be replaced by other sessions or otherPSKvalue.

The server accepts a ClientHello only if its binder validates, and the binder is a MAC over the truncated ClientHello under a key derived from the PSK, so the key shares, the ticket identity and the obfuscated age cannot be changed without detection. The parameters of the resumption ServerHello are covered by the same argument, because the handshake key both sides derive fixes the whole transcript.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value medium: RFC 9846 requires the binder to be validated before PSK key establishment and states that the Finished MAC prevents downgrade; the model shows the substitution fails.
- Stated in the documents: yes. Where the documents say it: rfc9846:6499
- Rests on: binder and transcript modelled as MAC and hash chains over public fields; only the X25519MLKEM768 group is negotiated, so a downgrade to another group is not modelled
- Next step: none

K01: secrecy of PSK and traffic keys

The assailant cannot extrapolate resumptionPSK、binder_key, or apply the flow key.

The answers of this target changed after this reading was written.

The PSK, the binder key and the application traffic keys of both connections stay out of reach of an attacker who only controls the network. He would have to invert the KDF or obtain a key share secret, and neither is possible in the model.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value medium: RFC 9846 Appendix F.1 states that the shared session keys should be known only to the communicating parties; this is what the queries confirm.
- Stated in the documents: yes. Where the documents say it: rfc9846:6483
- Rests on: the KDF and the transcript hash are one-way functions the attacker cannot apply to known inputs; X25519 by g^xy = g^yx only; the KEM is deterministic and its decapsulation is keyed by the client's seed
- Next step: none

C01: PSK leak spares earlier session

The release PSK is leaked after the resumed session has been completed and does not expose previously completed session keys or applied data.

A resumption PSK learnt after the original connection has completed does not expose that connection's application traffic key. The PSK comes from the resumption secret, which descends from the master secret in one direction only, so the traffic key cannot be recomputed from it.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value medium: RFC 9846 Appendix F.1 states that the resumption secret of connection N is separate from the traffic keys of connection N, so a later compromise does not reach them.
- Stated in the documents: yes. Where the documents say it: rfc9846:6504
- Rests on: the PSK is leaked after the run, in a separate phase; the KDF is one-way; no traffic key is a function of the PSK alone

- Next step: none

C03: resumption key isolated from PSK and original key

divulges the flow key for resumption PSK or for the original session, without exposing the application flow key for the resumed session.

The answers of this target changed after this reading was written.

Neither the resumption PSK nor the original connection's application traffic key gives the resumption connection's application traffic key. The resumption master secret mixes the PSK with a fresh hybrid shared secret and the new transcript, so a known PSK or a known earlier key does not reproduce it.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value medium: RFC 9846 Appendix F.1 describes exactly this separation between the connections; the proofs confirm it for the profile.
- Stated in the documents: yes. Where the documents say it: rfc9846:6551
- Rests on: the resumption key shares are freshly generated and never leaked in these queries; the KDF is one-way
- Next step: none

C04: one broken KEX component spares keys

The leak is back in hand.X25519Temporary private key orML-KEMOne of the private keys, resume the session key remains confidential.

Leaking the client's X25519 ephemeral scalar, or the ML-KEM decapsulation key of the resumption, does not give the resumption session keys, because the handshake secret is extracted from the concatenation of both key exchange results together with the PSK and the transcript, and no single component determines it. This is the hybrid property the profile depends on.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value medium: RFC 9954 Section 6 requires the hybrid shared secret to remain secure while one component key exchange is unbroken, and RFC 10024 relies on it for X25519MLKEM768.
- Stated in the documents: yes. Where the documents say it: rfc9954:550
- Rests on: each leaked value is given after the run; the ML-KEM query also gives the PSK and the X25519 scalar; the hybrid secret is the plain concatenation of the two component secrets, without a length prefix or further mixing
- Next step: none

S04: resumption inherits the original context

Restores the identity of the session, PSK and the context of the key correctly inherits from the original handshake and is not misdirected to other sessions.

The PSK a client offers in a resumption is one it received in a NewSessionTicket of a handshake that had completed, and the ticket and PSK were derived from that connection's resumption secret, so the resumption inherits the original context instead of another session's. The client also binds the ticket to the server it authenticated.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value low: RFC 9846 Appendix F.1 states that the binder binds the PSK to the session in which it was established, which is what the query confirms by structure.

- Stated in the documents: yes. Where the documents say it: rfc9846:6585
- Rests on: one ticket per server run; the server derives the PSK itself, so no ticket lookup across sessions is modelled
- Next step: none

A08.9: PSK and binder tied to server and transcript

Restoring the accepted PSK/binder binding to the original server identity and restoration of transcript.

The binder accepted by the server is the MAC of exactly the ClientHello the client sent, so the PSK is tied to the resumption transcript, and the client accepts a ticket only in a session in which it verified the server's certificate and CertificateVerify. A ticket therefore stays attached to the server that issued it and to the handshake it came from.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value medium: RFC 9846 Appendix F.1 states the binder binds the PSK to the current handshake and to the session that established it, and warns that a PSK shared with a second endpoint is unsafe.
- Stated in the documents: yes. Where the documents say it: rfc9846:6585, rfc9846:6600
- Rests on: a single pinned server certificate key; one client and one server process
- Next step: two servers accepting each other's tickets would test the PSK-FINISHED warning directly

U01: single leak leaves resumption key secret

Separate DisclosurePSKOr a new one.KEXScale, whether the restoration key remains confidential.

The answers of this target changed after this reading was written.

Any single leak among the resumption PSK, the client's X25519 ephemeral scalar and the ML-KEM decapsulation key leaves the resumption session key secret. Because each query also hands the attacker the earlier rungs of the ladder, the answers cover more than one leaked value at a time.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value medium: RFC 9846 Appendix F.1 and RFC 9954 Section 6 state these limits of a leak in one direction; the model confirms them for the resumption.
- Stated in the documents: yes. Where the documents say it: rfc9846:6562, rfc9954:550
- Rests on: leaks are given after the run, in separate phases; the KEM is deterministic and its shared secret is a fresh value chosen by the server; the PSK-only query sits at a phase where the X25519 scalar is not yet known, but ProVerif's phase counting also admits the later leaks
- Open: each query also includes the earlier leaks of the ladder, so the answers are stronger than a single leak
- Next step: separate models per leak would answer each leak with exactly one leaked value

C03.11: application keys isolated between connections

Restoring the session application key is isolated from the original session application key, and the leaking party does not expose the other party.

The original connection's application traffic key does not give the resumption connection's application traffic key, and the resumption key does not give the original connection's server-to-client key. The two keys descend from different master secrets through one-way derivations.

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

- Value medium: RFC 9846 Appendix F.1 states the separation between the connections; both directions that the model can hold at once are proved.
- Stated in the documents: yes. Where the documents say it: rfc9846:6551
- Rests on: the second direction is answered for the server-to-client key of the original connection, which the model never leaks; the KDF is one-way
- Next step: none

X01: distinct handshakes give distinct session keys

No two distinct sessions derive the same resumption application traffic key: if the client's session and the server's session name the same key, the two sessions are the same session.

Two resumption sessions that name the same application traffic key are the same session. Fresh hybrid key shares in each run are what makes the keys distinct.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value low: RFC 9846 Appendix F.1 asks that distinct handshakes produce distinct, unrelated session keys.
- Stated in the documents: yes. Where the documents say it: rfc9846:6495
- Rests on: the identity of a session is its final transcript hash; one client and one server process
- Next step: none

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
A03	proved		a03_srv	checked		2 proved
A04	proved		a04_inj	checked		2 proved
A08	proved		a08_sub	checked		1 proved
K01	proved		k01_psk	checked		5 proved
C01	proved		c01	checked		1 proved
C03	proved		c03	checked		3 proved
C04	proved		c04_x	checked		2 proved
S04	proved		s04	checked		1 proved
A08.9	proved		a08_sub	checked		2 proved
U01	proved		u01_psk	checked		4 proved
C03.11	proved		c03	checked		2 proved
X01	proved		x01	checked		1 proved

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

target	claim	attacker	answer	query	control
A03	server side agreement: when the server accepts the resumption (the client Finished verifies), a client ran the resumption with the same PSK and the same handshake key hs2, which fixes the transcript, the selected PSK identity and both key shares	`network`	proved	a03_srv	checked
A03	client side agreement: when the client accepts the server's resumption Finished, the server sent it under the same PSK and the same handshake key hs2, so both sides agree on the PSK identity, the key shares and the transcript-derived keys	`network`	proved	a03_cli	checked
A04	injective agreement on the resumption: each acceptance by the server of a client Finished corresponds to a distinct client run; the same Finished is not accepted twice in the modelled exchange	`network`	proved	a04_inj	checked
A04	injective agreement on the resumption: each acceptance by the server of a client Finished corresponds to a distinct client run; the same Finished is not accepted twice	`network`	proved	a04_inj	checked
A08	the ClientHello a server accepts with a valid binder is exactly one a client sent: the binder and both key shares cannot be substituted	`network`	proved	a08_sub	checked
K01	the resumption PSK is known to no one but the two parties, against the attacker on the network alone	`network`	proved	k01_psk	checked
K01	the binder key derived from the PSK is secret against the network attacker	`network`	proved	k01_binder	checked
K01	the application traffic key of the original connection is secret against the network attacker	`network`	proved	k01_orig	checked

target	claim	attacker	answer	query	control
K01	the handshake and application traffic keys of the resumption connection are secret against the network attacker	`network`	proved	k01_res	checked
K01	the application traffic keys of the resumption connection (handshake and application) are secret against the network attacker	`network`	proved	k01_res	checked
C01	the application traffic key of the original connection stays secret when the attacker holds the resumption PSK learnt after that connection completed	`psk_leak`	proved	c01	checked
C03	the resumption connection's application traffic key stays secret when the attacker holds the resumption PSK (and, in this query, also both private halves of the resumption key share)	`psk_leak`	proved	c03	checked
C03	the resumption connection's application traffic key stays secret when the attacker holds the original connection's application traffic key	`orig_app_key_leak`	proved	c03	checked
C03	the resumption connection's application traffic key stays secret when the attacker holds the resumption PSK (and, in this query, also the X25519 scalar and the ML-KEM decapsulation key of the resumption)	`psk_leak`	proved	c03	checked
C04	the resumption application traffic key stays secret when the attacker holds the client's X25519 ephemeral scalar of the resumption key share (and, in this query, also the resumption PSK)	`x25519_scalar_leak`	proved	c04_x	checked
C04	the resumption application traffic key stays secret when the attacker holds the ML-KEM decapsulation key of the client's resumption key share (and, in this query, also the PSK and the X25519 scalar)	`mlkem_sk_leak`	proved	c04_kem	checked
S04	the PSK a client uses in a resumption is one it received in a NewSessionTicket of a completed handshake, so the resumption inherits the original context and not another session's	`network`	proved	s04	checked
A08.9	the binder a server accepts binds the exact ClientHello (truncated transcript) the client sent, so the PSK and binder are tied to the resumption transcript	`network`	proved	a08_sub	checked
A08.9	a client accepts a PSK in a NewSessionTicket only in a session where it verified the server's certificate key and the CertificateVerify signature, so the PSK is tied to the original server identity	`network`	proved	a089_srv	checked
U01	the resumption session key stays secret when the X25519 ephemeral scalar of the resumption leaks (the attacker in this query also holds the PSK)	`x25519_scalar_leak`	proved	c04_x	checked
U01	the resumption session key stays secret when the ML-KEM decapsulation key of the resumption leaks (the attacker in this query also holds the PSK and the X25519 scalar)	`mlkem_sk_leak`	proved	c04_kem	checked
U01	the resumption session key stays secret when the resumption PSK is leaked	`psk_leak`	proved	u01_psk	checked
U01	the resumption session key stays secret when exactly the resumption PSK is leaked	`psk_leak`	proved	u01_psk	checked
C03.11	leaking the resumption connection's application traffic key does not expose the original connection's application traffic key (the server-to-client direction, which is never leaked)	`res_app_key_leak`	proved	c0311b	checked
C03.11	leaking the original connection's application traffic key does not expose the resumption connection's application traffic key	`orig_app_key_leak`	proved	c03	checked
X01	two sessions that derive the same resumption application traffic key are the same session	`network`	proved	x01	checked

Appendix D: How the analysis was run

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

ProVerif+ Agent session session-20261115T080741, 2026-11-15. Status: finished. Model: deepseek-v4-1-flash-261017. Steps: 66, prover 311 s, wall 19 min, cost 0.158 USD.

Context: the host removed stale parts from the conversation 1 time(s), 453010 characters in all (earlier reasoning, and what showed the model before its latest change). The trajectory file keeps everything.

ProVerif: 93 proofs run, 93 decided, 0 without an answer, 230 s, 45 answers recorded.

What was analysed

Protocol	TLS hybrid PSK-DHE resumption profile
Version	rfc10024
Analysed	One full time.X25519MLKEM768Shake hands get resumptionPSKAgain.psk_dhe_keRehabilitation; Keep ticket age/identity, binder, transcript and newkey_share； _Other Organiser0-RTT, ticket cross-cluster distribution, anti-replacement storage assurance.
Not analysed	Symbolic models do not support the calculation of algorithms as safe or as resistant to quantum security; The trajectory of the attack is established only under clear assumptions and is not directly equivalent to a normative loophole. Each nature must be subject to accessibility/proportional controls.
Source text	512 numbered passages (135022 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
rfc9846	4	Handshake Protocol
rfc9846	4.1	The Transcript Hash
rfc9846	4.2	Key Exchange Messages
rfc9846	4.2.1	Cryptographic Negotiation
rfc9846	4.2.2	Client Hello
rfc9846	4.2.3	Server Hello
rfc9846	4.2.4	Hello Retry Request
rfc9846	4.3	Extensions
rfc9846	4.3.1	Supported Versions
rfc9846	4.3.2	Cookie
rfc9846	4.3.3	Signature Algorithms

Document	Section	Title
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
rfc9846	7.5	Exporters

The promise of the program as the source step listed it in the documents:X25519MLKEM768Shake hands get resumptionPSKAgain.psk_dhe_keRehabilitation; Keep ticket age/identity, binder, transcript and newkey_share; _Other Organiser0-RTT, ticket cross-cluster distribution, anti-replacement storage assurance.

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
A03	resumption session key agreement	client and server agree on the PSK identity, the key shares, the transcript and the handshake keys they derive in the psk_dhe_ke resumption	RFC 9846 Appendix F.1 requires that the handshake outputs the same set of session keys on both sides when it completes on each endpoint.	rfc9846:6478
A04	no replay of a resumption flight	every accepted resumption corresponds to a distinct peer run; one ClientHello with its binder, or one Finished, is not accepted twice	RFC 9846 Appendix F.1 asks for unique session keys between distinct handshakes, and Section 4.5.3 requires a recipient to reject a Finished that does not verify.	rfc9846:6495, s0399
A08	resumption parameters not substitutable	the group, key shares, selected PSK identity and transcript accepted in a resumption are those of this session, not of another session or of another ticket	RFC 9846 asks for downgrade protection (the cryptographic parameters the same on both sides) and states that the Finished MAC prevents downgrade; the binder binds the PSK to the current handshake.	rfc9846:6499, rfc9846:6627, rfc9846:6585
K01	secrecy of PSK and traffic keys	the resumption PSK, the binder key, and the handshake and application traffic keys are known only to the two parties	RFC 9846 Appendix F.1: the shared session keys should be known only to the communicating parties and not to the attacker.	rfc9846:6483
C01	PSK leak spares earlier session	a resumption PSK learnt after the connection has completed does not expose the keys or the application data of the original connection	RFC 9846 Appendix F.1 treats the resumption PSK as long-term keying material and requires forward secrecy with respect to it, and states that the resumption secret of connection N is separate from the traffic keys of connection N.	rfc9846:6504, rfc9846:6551
C03	resumption key isolated from PSK and original key	the application traffic key of the resumption connection, when the attacker knows the resumption PSK or the original connection's application traffic key	RFC 9846 Appendix F.1: the resumption secret needed for connection N+1 is separate from the traffic keys of connection N, so forward secrecy between the connections holds, and compromise at an earlier time does not give keys at a later time when asymmetric key establishment is rerun.	rfc9846:6551, rfc9846:6562
C04	one broken KEX component spares keys	the resumption session keys, when the attacker holds either the X25519 ephemeral scalar or the ML-KEM decapsulation key of the resumption key share	RFC 9954 Section 6: the hybrid shared secret is secure as long as at least one of the component key exchange algorithms is unbroken; RFC 10024 relies on this for X25519MLKEM768.	rfc9954:550, s0015, s0018
S04	resumption inherits the original context	the ticket identity, the PSK and the key context a resumption uses are those of the original handshake, not those of another connection or another ticket	RFC 9846 Appendix F.1: the binder forms a binding between the PSK, the current handshake and the session in which the PSK was established, transitively including the original handshake transcript.	rfc9846:6585

ID	Name	About	What the documents say	Where
A08.9	PSK and binder tied to server and transcript	the PSK and binder a resumption accepts are tied to the server identity that issued the ticket and to the resumption transcript	RFC 9846 Appendix F.1 states the binder binds the PSK to the current handshake and to the session that established it, and warns that a PSK shared with a second endpoint is unsafe (the PSK-FINISHED construction).	rfc9846:6585, rfc9846:6600
U01	single leak leaves resumption key secret	the resumption session keys, once the attacker holds exactly one of the resumption PSK, the client's X25519 ephemeral scalar, or the client's ML-KEM decapsulation key	RFC 9846 Appendix F.1 fixes what a leak in one direction gives (compromise at T1 does not give keys at T2 when asymmetric key establishment is rerun) and RFC 9954 Section 6 gives the hybrid requirement.	rfc9846:6562, rfc9954:550
C03.11	application keys isolated between connections	the application traffic keys of the original connection and of the resumption connection, one against a leak of the other	RFC 9846 Appendix F.1: the resumption secret of connection N is separate from the traffic keys used by connection N, providing forward secrecy between the connections.	rfc9846:6551, rfc9846:6573
X01	distinct handshakes give distinct session keys	the client and the server, about the application traffic key of the resumption handshake	RFC 9846 Appendix F.1: any two distinct handshakes should produce distinct, unrelated session keys, and individual session keys produced by a handshake should be distinct and independent.	rfc9846:6495

Added from the documents by the analyst: X01.

The security statements of the documents, paragraph by paragraph

The host listed the 15 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: 3. Not taken: 12. 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	A03, A08, S04, A08.9	
P02	main:273-276	The prover cannot be asked	guidance on side-channel-resistant implementations; side channels and implementation quality are outside a symbolic protocol model
P03	main:278-280	The prover cannot be asked	fixed lengths of public keys, ciphertexts and shared secrets: a property of the encoding, not of any trace
P04	main:282-287	The prover cannot be asked	the ML-KEM encapsulation randomness is disclosed to the client and carries RNG output; the model idealises randomness and cannot speak of RNG state
P05	main:289-297	The prover cannot be asked	an attack that starts from the disclosure of an insecure RNG's output; randomness quality is not modelled
P06	main:299-306	C04, U01, `x25519_scalar_leak`	
P07	main:308-310	The prover cannot be asked	one insecure RNG shared by both algorithms: about RNG state, not modellable
P08	rfc9846:4676-4677	No goal stated	a pointer to Appendices C, E and F; states no property of its own
P09	rfc9954:550-558	C04, U01, `x25519_scalar_leak`, `mlkem_sk_leak`	

Paragraphs	Where	Taken up as	Why not
P10 to P15	rfc9954:560-597	The prover cannot be asked	IND-CCA2 security of the KEM, constant-length encodings, and timing or collision-resistance properties of the hash and KDF: cryptographic assumptions and implementation properties, none of which a symbolic trace model can decide

The parties and the protected values, one by one

The host listed the 23 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: 10. Without: 13. The list follows from the properties; what was made of each entry is the analyst's judgement.

ID	Party or value	Kind	Attacker	Why none
S01	Client	party	Nothing gained	a client that is corrupted holds only its own PSK and its own session keys; no target asks what other parties are assured of about a client, since this profile does not authenticate the client
S02	Servers	party	`psk_leak`	
S03	resumption PSK	value	`psk_leak`	
S04	binder	value	Nothing gained	the binder is sent in the clear in the ClientHello, so holding it gives nothing beyond the network attacker
S05	transcript	value	Nothing gained	the transcript is built from public handshake messages
S06	key_share	value	Nothing gained	the key shares are public; only their private halves matter and they are the entries S18 and S19
S07	Handshake key	value	Everything lost by definition	K01 only asks that the network attacker cannot derive the handshake traffic keys; once those keys are given, that is lost by definition and no document asks for protection against an attacker who already holds them
S08	Finished	value	Nothing gained	the Finished value is on the wire and is a MAC under a key the attacker does not have
S09	ticket identity	value	Nothing gained	the ticket identity travels in the clear as the PSK identity; no target asks for unlinkability of connections
S10	Consultative Group	value	Nothing gained	the negotiated group is public
S11	PSK identity	value	Nothing gained	the PSK identity is the ticket, sent in the clear
S12	binder_key	value	Everything lost by definition	K01 asks that the binder key cannot be derived by the network attacker; giving it away loses that by definition and no document asks for more
S13	Apply flow keys	value	`orig_app_key_leak`, `res_app_key_leak`	
S14	Session Key	value	`orig_app_key_leak`, `res_app_key_leak`	
S15	Application data	value	`orig_app_key_leak`, `res_app_key_leak`	
S16	Original Session Apply Flow Key	value	`orig_app_key_leak`	
S17	Restore Session Application Flow Key	value	`res_app_key_leak`	

ID	Party or value	Kind	Attacker	Why none
S18	X25519 temporary private key	value	`x25519_scalar_leak`	
S19	ML-KEM private keys	value	`mlkem_sk_leak`	
S20	Restore Session Key	value	Everything lost by definition	C04 and U01 ask that the resumption session keys stay secret when one key exchange component leaks; an attacker who holds the session keys themselves loses that by definition, and no document asks for it
S21	PSK	value	`psk_leak`	
S22	Context of Session	value	Nothing gained	the session context consists of public handshake messages
S23	Server Identity	value	Nothing gained	the server's certificate is public

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		
`psk_leak`	the resumption PSK of a connection, learnt from the server's ticket store or after the connection completed; the resumption PSK is long-term keying material that outlives the connection	RFC 9846 Appendix F.1 treats the external or resumption PSK as long-term keying material whose compromise after the handshake must not break the session key, and discusses tickets stored in a database.	rfc9846:6504, rfc9846:6547
`x25519_scalar_leak`	the X25519 ephemeral private scalar of the resumption handshake key share, obtained from the endpoint or recovered by a passive observer with a cryptographically relevant quantum computer	RFC 10024 Section 6 states that any passive observer with a quantum computer can recover the ECDH ephemeral scalar, and RFC 9954 Section 6 requires the hybrid secret to stay secure when one component is broken.	main:299, rfc9954:550
`mlkem_sk_leak`	the ML-KEM decapsulation key belonging to the client's key share in the resumption handshake	RFC 9954 Section 6 requires the hybrid shared secret to remain secure as long as one of the two component key exchanges is unbroken, which is what this leak tests.	rfc9954:550, s0015
`orig_app_key_leak`	the application traffic key of the original connection, obtained after that connection completed	RFC 9846 Appendix F.1 limits the effect of key leakage in one direction and states which protection a rerun of asymmetric key establishment gives after a traffic secret is compromised.	rfc9846:6562, rfc9846:6573
`res_app_key_leak`	the application traffic key of the resumption connection, obtained after it completed	RFC 9846 Appendix F.1 states that compromise of a key at a later time does not give some key at an earlier time.	rfc9846:6562

ID	`network`	`psk_leak`	`x25519_scalar_leak`	`mlkem_sk_leak`	`orig_app_key_leak`	`res_app_key_leak`
A03	Holds					
A04	Holds					

ID	network	psk_leak	x25519_scalar_leak	mlkem_sk_leak	orig_app_key_leak	res_app_key_leak
A08	Holds					
K01	Holds					
C01		Holds				
C03		Holds			Holds	
C04			Holds	Holds		
S04	Holds					
A08.9	Holds					
U01		Holds	Holds	Holds		
C03.11					Holds	Holds
X01	Holds					

Notes on the prover and the model

What the analyst named for targets:

step	targets	features	reason
55	A03, A04, A08, K01, C01, C03, C04, S04, A08.9, U01, C03.11, X01	plain_primitives, other	Every answer rests on hashing, MAC, KDF, a KEM and Diffie-Hellman used only as a plain key agreement with the single law g^xy = g^yx; the KEM is deterministic (no encapsulation randomness) and the transcript hash functions are private, so the attacker can neither compute transcript hashes of his own nor exploit a distributed KEM. The model runs one client and one server process without replication, so the replay and cross-session answers (A04, X01, S04) hold for that single run only and say nothing about a server that keeps tickets across sessions.

The analyst's closing note

The model is one full X25519MLKEM768 handshake followed by one psk_dhe_ke resumption (model.pv, one client and one server process, 26 events, normal run found). All twelve targets are answered, every one by a ProVerif proof and none falsified, and each proof carries a control query that the model falsifies, so no finding arises.

Established: both sides agree on the PSK identity, the key shares, the transcript and the derived keys (A03, both directions); the binder cannot be substituted, a ticket is only taken after the certificate and CertificateVerify check, and a resumption PSK always comes from a completed handshake (A08, A08.9, S04); distinct sessions give distinct application keys (X01) and each server acceptance matches a distinct client run (A04, within one session). Secrecy of the PSK, the binder key and the traffic keys against the network attacker holds (K01). A resumption PSK learnt afterwards does not expose the original connection's key (C01), a PSK leak or a leaked original application key does not give the resumption key (C03), and neither the X25519 scalar nor the ML-KEM decapsulation key alone gives the resumption keys, even together with the PSK (C04, U01) — the hybrid property RFC 9954 asks for. Application keys of the two connections are isolated in both directions the linear leak ladder can hold (C03.11).

What blocks full answers: the model runs one client and one server without replication and keeps no ticket store, so replay into a second session and reuse of one ticket (A04, and the cross-session part of X01 and S04) are outside it; the KEM is deterministic, the transcript and KDF hashes are private functions the attacker cannot apply, X25519 has only the law g^xy = g^yx, and the ticket age and lifetime are carried but not checked. The leaks are placed after the run in separate phases, and each query also holds the earlier leaks of the ladder, which makes the answers stronger than a single leak but couples them. A prover that models a server keeping tickets across sessions is needed for the replay questions; no target rests on a changing counter, so no bounded model and no state review were required.