Security analysis of SSH composite host authentication

This report in another language: 中文.

draft-ietf-sshm-composite-sigs-00 · ProVerif+ Agent report

Documents	Post-Quantum Composite Signatures in SSH (protocol definition); Post-Quantum/Traditional Hybrid Key Exchange with the Module-Lattice-Based Key-Encapsulation Mechanism for Use in SSH (supporting); The Secure Shell (SSH) Transport Layer Protocol (supporting)
Properties examined	13: 13 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

SSH composite host authentication in this draft lets a client check who the server is with a host key that combines ML-DSA-44 and Ed25519, while the connection key is agreed with both X25519 and ML-KEM-768. The client C holds the server's composite host key K_S in a trusted known-hosts entry; the server S holds the matching composite private key. The exchange runs a KEXINIT round and then the hybrid key exchange init and reply, in which S signs the exchange hash with both components and C checks both.

When the exchange ends the client is sure that it talked to the holder of the pinned composite host key, that the exchange hash it verified covers both algorithm name-lists, both identification strings, the host key, both key exchange octet strings and the shared secret, and that the session keys come from a secret no one on the network has. The server believes only that it computed a hash itself and signed it; in the modelled run nothing that follows uses the new keys.

The parties:

- **C**: client, verifies the composite host signature. Holds: known-hosts entry for S: composite host key K_S (mldsa44_pk, ed25519_pk).
- **S**: server, signs the exchange hash with both components. Holds: composite host private key (mldsa_seed, ed25519_seed).

Figure 1 shows the exchange as the documents specify it.

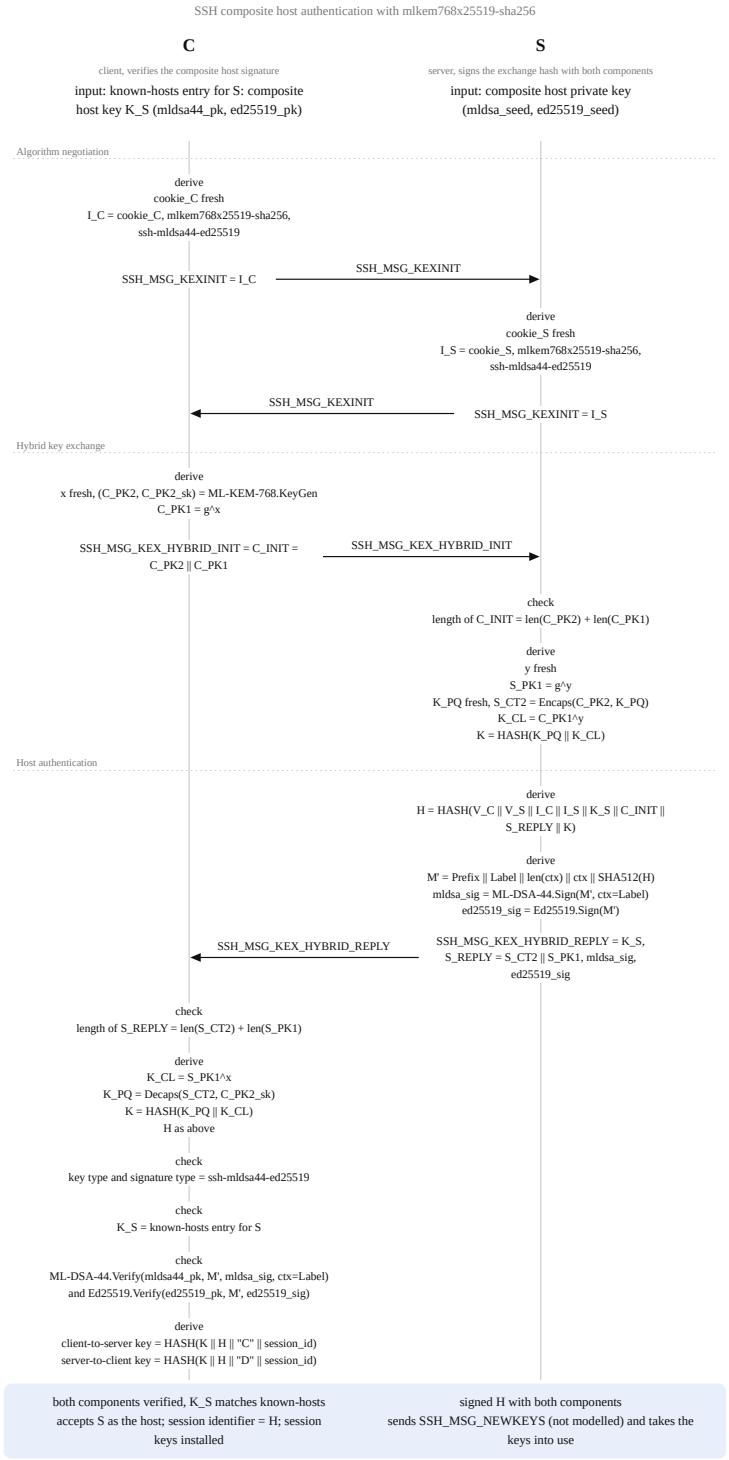


Figure 1. The protocol as specified.

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.

No finding: within the modelled flow the draft's composite host authentication gives what it is meant to give, and nothing appeared that the documents do not already state. The client is assured of the server's identity, of the exchange hash and all its fields including both algorithm name-lists, of the pinning of the host key to this session, and of the session keys; each accept matches a distinct server run. Secrecy holds for the host private key, the session keys and the exchange hash, and it survives reading the host private key from storage after the connection. Breaking one of the two component algorithms does not give host authentication, which is the dual-algorithm promise the draft makes. These are proofs about a symbolic model: a broken component is modelled as a forgery capability, X25519 by one Diffie-Hellman law, and rekey, user authentication and the length checks of the hybrid messages are left out. Twelve of the thirteen targets are answered in full and one only in part, the key-reuse half of key separation, which is a rule for generating keys rather than a property of a run.

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	**agreement on exchange hash and shared secret.** Whenever the client accepts server host authentication (the composite signature verifies and the session keys are accepted), there is a real run of the server, and the two agree on the client/server identities, the host public key K_S, the fields of the exchange hash (including the negotiated algorithm name-lists) and the shared secret K.	no compromise	Holds	medium	yes
A02	**server authentication to the client.** Whenever the client accepts host authentication, there is an instance of the protocol running as that server identity.	no compromise	Holds	medium	yes
A08	**negotiated algorithms bound in the signature.** In the exchange hash covered by the host signature the client accepts, the algorithm name-lists, the client/server identifiers and the host public key match the result of this negotiation, and signature verification uses the host key algorithm chosen by the negotiation.	no compromise	Holds	medium	yes
A04	**distinct server run per accepted session.** Each accept of host authentication by the client corresponds to a different run of the server (injective), and a signed exchange from an old session cannot be replayed as a new session.	no compromise	Holds	medium	yes
K01	**secrecy of host keys and session keys.** The composite host private key (mldsa_seed, ed25519_seed or d) and the session keys derived from K and H cannot be derived by an attacker who does not hold them.	no compromise	Holds	medium	yes
C08	**composite forgery needs both components.** If one of the ML-DSA component (attacker holds mldsa_sk) or the classical component (attacker holds ed25519_sk/ecdsa_sk) is broken, the attacker still cannot forge a composite host signature that passes verification, i.e. the client will not accept an exchange hash that no server issued.	a single component of the server's composite host signing key (the attacker already holds that component's private key before signature verification)	Holds	medium	yes

ID	Property	Compromise assumed	Result	Value	In the documents
K04	**keys distinct per purpose and direction.** Component private keys are not reused between different composite host keys or between composite and non-composite keys; the encryption/integrity keys for the client-to-server and server-to-client directions differ.	no compromise	Holds	low	yes
D01	**no man in the middle host substitution.** The exchange hash and host public key the client accepts were indeed issued by the claimed server and were not replaced by a man in the middle.	no compromise	Holds	medium	yes
U01	**one broken component does not break host auth.** When one component private key of the composite host signature leaks, the host authentication the client accepts still corresponds to a real server run, and the attacker cannot forge the composite signature.	a single component of the server's composite host signing key (before signature verification)	Holds	medium	yes
A08.10	**host key bound to the session.** The host signature the client accepts covers this session's exchange hash (including V_C, V_S, I_C, I_S, K_S, C_INIT, S_REPLY, K), so the host public key is bound to this session.	no compromise	Holds	medium	yes
U02	**both signature components must verify.** When the client accepts host authentication, both components of the composite signature verify successfully, and the negotiated algorithm matches the signature format, so an attacker cannot bypass either component by substitution.	no compromise	Holds	medium	yes
X01	**secrecy of the exchange hash.** An attacker who has only the network cannot derive the exchange hash H of a connection in which the client accepted the host, i.e. H stays unknown to him.	no compromise	Holds	low	yes
X02	**distinct session identifiers per session.** Two accepted sessions never share the same exchange hash H (session identifier), so a message bound to one session cannot be replayed into another.	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, X02.

The questions asked, and the analyst's answers:

Question	Answer	Rests on
Q1: host authentication when one component signing key leaks	No: with the ML-DSA component broken, or with the classical component broken, the client accepts no host signature that no server run produced, because both component verifications are required, so the loss of one component alone does not give host authentication to the attacker.	C08, U01, A02
Q2: binding of the host key to the session identifier/exchange hash	Yes: the host key enters the exchange hash together with the identification strings, both KEXINIT payloads, both key exchange values and the shared secret, so the accepted key is bound to this session and to its session identifier and cannot be taken from another session.	A08.10, A03, X02, A04
Q3: algorithm negotiation/encoding substitution does not bypass the other signature component	No: a rewritten name-list, key type or signature type name changes the exchange hash and the reply is rejected, and a reply that carries only one valid component is rejected because both verifications must succeed.	A08, U02

4 Limits of this analysis

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

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

- The computational security of ML-DSA-44, Ed25519 and ML-KEM is not shown; the model treats them as ideal primitives and represents a broken component as an explicit forgery capability for that component.
- X25519 is modelled with the single law g^xy = g^yx and ML-KEM as a free encapsulation function; other algebraic properties of the curve and the encodings are not represented.
- The length checks on C_INIT and S_REPLY that RFC 10042 requires against length-extension attempts are not modelled, because a symbolic pairing has no ambiguous concatenation.
- Strong unforgeability is not modelled: signatures are deterministic terms, so a second valid signature for the same exchange hash cannot arise and the documented SUF-CMA failure cannot be reproduced.
- No rekeying, no user authentication and no encrypted data channel are modelled, so nothing is said about re-exchange, about user authentication methods or about how the keys are used.
- The host key is assumed to be pinned in known-hosts before the connection; accepting an unverified host key is outside the analysis.

What the host records:

- The answers of A03, A04, K01, X01, X02 rest on an operation the model simplifies (Appendix D).

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: agreement on exchange hash and shared secret

Whenever a client accepts server host authentication (the composite signature verifies and the session key is accepted), the server really ran the exchange, and both sides agree on the client/server identities, the host public key K_S, every field of the exchange hash (including the negotiated algorithm name-lists) and the shared secret K.

Whenever the client accepts host authentication, the server really ran the exchange with the same host key, the same exchange hash and the same shared secret. The hash covers both algorithm name-lists, both identification strings, the host key, both key exchange octet strings and the shared secret, so an attacker who alters any of them makes the client reject the reply.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value medium: RFC 4253 and RFC 10042 state that the exchange hash authenticates the key exchange and list its fields; the result confirms that the hybrid variant keeps that.
- Stated in the documents: yes. Where the documents say it: rfc4253:1428, s0217, s0144
- Rests on: X25519 is represented by one Diffie-Hellman law, ML-KEM by a free encapsulation function and the hash by an ideal function.; The exchange hash binds the KEXINIT payloads as opaque strings; the model does not parse the name-lists inside them.; Known-hosts pinning is assumed, and user authentication and rekey are not modelled.
- Next step: Nothing open for the modelled flow; a model that parses the name-lists would be needed to speak about which algorithm was selected inside them.

A02: server authentication to the client

Whenever the client accepts host authentication, there is a protocol instance running as that server.

A client that accepts host authentication is talking to the holder of the composite host key pinned for the server, not to an attacker on the network. Only a party with both component private keys can sign the client's fresh exchange hash.

- Result: Holds; answers what is asked: yes; usable
- Value medium: RFC 4253 says the key exchange uses explicit server authentication and that the client verifies the host key against a local database; RFC 4251 says the transport performs server host authentication.
- Stated in the documents: yes. Where the documents say it: rfc4253:1428, rfc4251:652, s0154, s0209
- Rests on: Both signature components are ideal schemes and the host key is pinned before the connection.; The model has one composite key per server; several servers or several keys per name are not represented.
- Next step: Nothing open.

A08: negotiated algorithms bound in the signature

In the exchange hash covered by the host signature the client accepts, the algorithm name-lists, the client/server identifiers and the host public key match the result of this negotiation, and signature verification uses the host key algorithm chosen by the negotiation.

The signature the client accepts covers the KEXINIT payloads both sides exchanged, so an attacker who rewrites the server's name-list, or who moves the exchange to another host key algorithm, makes the two exchange hashes differ and the reply is rejected. The client also refuses a reply whose key type or signature type name is not the negotiated composite algorithm.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value medium: RFC 10042 lists I_C and I_S among the fields of the hybrid exchange hash and RFC 4253 says the exchange hash covers both KEXINIT payloads; the result confirms the binding holds for the composite host signature.
- Stated in the documents: yes. Where the documents say it: rfc4253:1428, s0144, s0169
- Rests on: Both KEXINIT payloads are modelled as opaque strings that enter the hash; the negotiated selection is not computed from the name-lists.; Guessed initial packets (first_kex_packet_follows) are not modelled.
- Open: Whether the negotiated selection itself, rather than the whole KEXINIT payloads, is compared with the key type is a matter for implementations; the model checks the type names but no selection algorithm.
- Next step: Model the name-list selection explicitly to speak about a downgrade between two composite algorithms.

A04: distinct server run per accepted session

Every accept of host authentication by the client corresponds to a different run of the server (injectivity), and the signature exchange of an old session cannot be replayed as a new session.

Every accept of host authentication matches a different run of the server, because the client contributes fresh values to the key encapsulation and to the X25519 share and both are inside the signed hash. A reply recorded in an old session is therefore not accepted in a new one.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value medium: RFC 4253 makes the first exchange hash the unique session identifier and RFC 4251 says public key authentication exchanges cannot be replayed in other sessions.
- Stated in the documents: yes. Where the documents say it: rfc4251:849, s0187
- Rests on: Freshness of the client's ephemeral X25519 secret and ML-KEM key is assumed, as the documents require.; Rekey is excluded, so the session identifier after a re-exchange is not examined.
- Next step: Nothing open.

K01: secrecy of host keys and session keys

The composite host private key (mldsa_seed, ed25519_seed or d) and the session keys derived from K and H cannot be derived by an attacker who does not hold them.

The composite host private key and the session keys stay unknown to an attacker on the network. Session keys also stay unknown when the whole composite host private key is read from the server's storage after the connection, because they are derived from the ephemeral key exchange and not from the host key; only a server that leaks its own session state gives them away.

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

- Value medium: RFC 4253 derives every encryption and integrity key from K and H, and RFC 4251 asks for forward secrecy of session keys and warns about collecting traffic for later decryption.
- Stated in the documents: yes. Where the documents say it: rfc4251:1021, s0189, s0197, s0138
- Rests on: Secrecy of the ephemeral X25519 secret and of the ML-KEM decapsulation result is the only protection of the shared secret.; Session keys are modelled as one hash derivation per direction; the extension procedure for long keys is not modelled.; The host key leak is placed after the modelled sessions have ended, which is the forward-secrecy reading.
- Open: Secrecy of the shared secret against a leak of one ephemeral value in the middle of a session is not asked.
- Next step: Tamarin with its equational theory could examine ephemeral-key compromise and the ML-KEM implicit rejection.

C08: composite forgery needs both components

When one of the ML-DSA component is broken (the attacker holds mldsa_sk) or the classical component is broken (the attacker holds ed25519_sk/ecdsa_sk), the attacker still cannot forge a composite host signature that passes verification, that is, the client will not accept an exchange hash the server did not issue.

Breaking one of the two component signature algorithms does not let an attacker authenticate as the server: the client requires both verifications, so with ML-DSA forgeable an accepted reply still needs an Ed25519 signature the attacker cannot produce, and the symmetric case follows the same code path. This is the dual-algorithm promise of the draft and it holds for the modelled exchange.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value medium: The draft states that an adversary must forge both ML-DSA and Elliptic Curve signatures and that the composite stays EUF-CMA while at least one component is unbroken; the result reproduces that mechanically for the SSH exchange.
- Stated in the documents: yes. Where the documents say it: main:447, main:455, s0096, s0098, s0083
- Rests on: A broken component is modelled as a universal forgery capability for that component, which is stronger than holding its private key; the classical case is the symmetric reading of the same query.; The computational statement that ML-DSA or Ed25519 is EUF-CMA is not proved; the model shows only that the composite fails when both halves fail.; The adversary class of the break (classical or quantum) is not represented.
- Open: The per-adversary-class degradation that the underlying specification describes is computational and outside the model.
- Next step: Nothing further in ProVerif; that claim needs a reduction proof, not a symbolic one.

K04: keys distinct per purpose and direction

Component private keys are not reused between different composite host keys, or between composite and non-composite keys; the encryption/integrity keys differ between the client-to-server and server-to-client directions.

The two directions of a connection always use different derived keys, so knowing one direction does not give the other. The other half of the target, that component key material is never reused outside its composite key, is a rule for how keys are generated and deployed, which a protocol model cannot show; the model enforces only that the pinned composite key type is accepted.

- Result: Holds; answers what is asked: partly; usable with caveats
- Value low: The draft forbids reusing component key material and RFC 4253 derives the two directions from different letters; the result confirms the derivation side only.
- Stated in the documents: yes. Where the documents say it: main:451, s0097, s0190, s0191

- Rests on: Key generation and deployment are outside the model; the reuse prohibition is taken as an assumption.; Key separation is shown only for the two modelled direction keys, not for the integrity keys or for several key lengths.
- Open: Key material reuse between composite and standalone keys cannot be asked of a protocol prover.
- Next step: An implementation or policy review would settle the reuse half.

D01: no man in the middle host substitution

The exchange hash and the host public key that the client accepts really were issued by the claimed server and were not replaced by a man in the middle.

A client with the server's composite host key pinned in known-hosts never accepts a host key or an exchange hash substituted on the network. Against an attacker without a component private key the signature over the client's fresh hash cannot be produced.

- Result: Holds; answers what is asked: yes; usable
- Value medium: RFC 4251 describes this man-in-the-middle case and says the attacker cannot succeed when the host key was verified beforehand, because he cannot sign the session-specific data.
- Stated in the documents: yes. Where the documents say it: rfc4251:905, s0209, s0219
- Rests on: The known-hosts entry is trusted and present before the connection; accepting a new host key is outside the model.; Signatures and hashes are ideal.
- Next step: Nothing open.

U01: one broken component does not break host auth

When one component private key of the composite host signature leaks, the host authentication the client accepts still corresponds to the server's real run, and the attacker cannot forge the composite signature.

One leaked component private key does not give host authentication to an attacker: the composite check needs a signature from the other component as well. The two readings, ML-DSA lost and classical lost, are answered by the same query because the model checks the two components independently and requires both.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value medium: The draft states that the composite remains unforgeable while one component is unbroken; this is the host authentication form of that statement.
- Stated in the documents: yes. Where the documents say it: main:447, s0096, s0101
- Rests on: The leak is modelled as the ability to forge that component rather than as possession of a key; this is the stronger capability, so the answer covers the key-leak reading.; No cross-protocol reuse of a component key is modelled, which the draft forbids.
- Open: The classical-component reading is answered by the symmetry of the model's two checks, not by a second query.
- Next step: A model with two distinct composite algorithms would answer both readings with separate queries.

A08.10: host key bound to the session

The host signature the client accepts covers the exchange hash of this session (including V_C, V_S, I_C, I_S, K_S, C_INIT, S_REPLY, K), so the host public key is bound to this session.

The composite host key the client accepts is inside the exchange hash of this very session, together with the client's fresh key exchange value, so a host key or host signature recorded in another session cannot be used here.

- Result: Holds; answers what is asked: yes; usable
- Value medium: RFC 10042 puts K_S into the hybrid exchange hash and RFC 4253 makes the first exchange hash the session identifier.
- Stated in the documents: yes. Where the documents say it: rfc4253:1428, s0144, s0187
- Rests on: The host key is modelled as the pair of public keys carried in the reply and inside the hash, as the wire formats require.; The exchange hash is taken as the session identifier of the first key exchange; rekey is excluded.
- Next step: Nothing open.

U02: both signature components must verify

When the client accepts host authentication, both components of the composite signature verify successfully and the negotiated algorithm matches the signature format, so an attacker cannot bypass either component by substitution.

The client accepts a composite host signature only when the ML-DSA-44 and the Ed25519 verification both succeed under the composite key type it negotiated. A reply that carries one valid component only, or that names another key or signature type, is rejected; the control run shows exactly that, where a client checking only ML-DSA accepts a reply forged with the broken component.

- Result: Holds; answers what is asked: yes; usable
- Value medium: The draft requires both component verifications to succeed and requires failure of either to fail the composite; the result confirms that a client implementing this is not bypassed by substitution.
- Stated in the documents: yes. Where the documents say it: main:440, s0083, s0088, s0077
- Rests on: The two component schemes are ideal, and the composite combiner with its prefix and label is represented by one function around the exchange hash.; The wire formats are represented by the algorithm type name in the key and in the signature blob, which the client compares with the negotiated name.
- Next step: Nothing open.

X01: secrecy of the exchange hash

An attacker who has only the network cannot derive the exchange hash H of a connection in which the client accepted the host, i.e. H stays unknown to him.

The exchange hash of an accepted connection stays unknown to an attacker on the network, because it includes the shared secret and both fresh key exchange values. The two documents pull apart here: RFC 4251 allows the session identifier to be public, while RFC 4253 asks for the exchange hash to be kept secret, and the secrecy reading is the one that holds of a real connection.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value low: RFC 4253 says the exchange hash should be kept secret, RFC 4251 says the session identifier can be public, and only the RFC 4253 reading is asked here.
- Stated in the documents: yes. Where the documents say it: rfc4253:218, rfc4251:849, s0218
- Rests on: The hash is ideal, so the hash hides its inputs; the claim would fail for a hash with leaked structure.; The hash is the session identifier and higher layers use it in ways the model does not represent.
- Open: Which of the two readings the documents intend is left open by the documents themselves.
- Next step: An editorial clarification of RFC 4253 and RFC 4251 would settle whether the session identifier is meant to be secret.

X02: distinct session identifiers per session

Two accepted sessions never share the same exchange hash H (session identifier), so a message bound to one session cannot be replayed into another.

Each accepted session matches a distinct run of the server and the client's own fresh key exchange values, so the exchange hash of one session is not reused by another accepted session. The model checks this as injective agreement, because its query language cannot state directly that two events carry different values.

- Result: Holds; answers what is asked: partly; usable with caveats
- Value low: RFC 4253 makes the exchange hash a unique identifier and RFC 4251 says two sessions sharing a session identifier would allow replay between them.
- Stated in the documents: yes. Where the documents say it: rfc4251:857, s0187
- Rests on: Uniqueness rests on the freshness of the client's ephemeral values and on the hash being ideal and injective; a hash collision is not modelled.; The statement is checked as injective agreement on host key, exchange hash and shared secret, not as a direct comparison of two session identifiers.
- Open: A direct no-collision statement is not expressible in the query language used.
- Next step: A prover with explicit equality or counting queries, such as Tamarin with restrictions, could state the collision property directly.

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		agree	checked		1 proved
A02	proved		agree	checked		1 proved
A08	proved		agree	checked		1 proved
A04	proved		inj	checked		1 proved
K01	proved		secret_sym	checked		3 proved
C08	proved		agree	checked		2 proved
K04	proved		dirsep	checked		1 proved
D01	proved		agree	checked		1 proved
U01	proved		agree	checked		2 proved
A08.10	proved		agree	checked		1 proved
U02	proved		agree	checked		1 proved
X01	proved		secret_h	checked		1 proved
X02	proved		inj	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	Whenever the client accepts host authentication, the composite host key K_S, the exchange hash H (which binds V_C, V_S, I_C, I_S, K_S, C_INIT, S_REPLY and K) and the shared secret K it accepted are those of one run of the server; the attacker has nothing beyond the network.	`network`	proved	agree	checked
A02	Whenever the client accepts host authentication there is a run of the server holding the composite host key pinned for it, i.e. the client is talking to the named server and not to the network attacker.	`network`	proved	agree	checked
A08	The exchange hash under the accepted host signature covers the name-lists I_C and I_S actually exchanged, the host key and the client's and server's key exchange values, and the client accepts only the composite algorithm name it negotiated; a rewritten name-list or key type gives a different H and is rejected. The attacker has nothing beyond the network.	`network`	proved	agree	checked
A04	Every accept of host authentication corresponds to a different run of the server, so a recorded key exchange reply cannot be replayed into a new session.	`network`	proved	inj	checked
K01	The session keys derived from K and H (the client-to-server and server-to-client keys) stay unknown to an attacker who has only the network; he learns them only for a session whose server leaks them.	`network`	proved	secret_sym	checked

target	claim	attacker	answer	query	control
K01	A session key stays secret even when the attacker reads the whole composite host private key from the server's storage after the connection: the session keys do not depend on the host key (forward secrecy).	`host_key_after_session`	proved	fs_sym	checked
K01	The composite host private key, both the ML-DSA component (mldsa_seed) and the Ed25519 component (ed25519_seed), stays unknown to an attacker who has only the network, during the key exchange.	`network`	proved	secret_priv	checked
C08	The ML-DSA-44 component is broken (the attacker can produce a valid ML-DSA component signature for any exchange hash): the client still accepts no composite host signature over an exchange hash that no server run signed, so host authentication survives the loss of that component.	`broken_component`	proved	agree	checked
C08	The same for the classical component: the model checks the two components by the same code path and requires both, so an attacker who can forge Ed25519 component signatures only is in the symmetric position to the one above and is equally unable to authenticate as the server.	`broken_component`	proved	agree	checked
K04	The client-to-server and the server-to-client key of one session are always different values, so knowing one direction does not give the other and the two directions are separated by their derivation inputs.	`network`	proved	dirsep	checked
D01	A client that holds the composite host key of the server from known-hosts never accepts a host key or exchange hash substituted by the network attacker.	`network`	proved	agree	checked
U01	With one component of the composite host signature broken, an attacker still cannot make the client accept host authentication for the server without a run of that server (ML-DSA broken; the classical case is the symmetric reading of the same query).	`broken_component`	proved	agree	checked
U01	The classical component is broken and the ML-DSA component still holds: the same query covers it, because the two component checks are independent and both are required, so the attacker cannot authenticate as the server (symmetric reading).	`broken_component`	proved	agree	checked
A08.10	The composite host key the client accepted is bound to this session's exchange hash and session identifier, so a host key taken from another session cannot be used in this one.	`network`	proved	agree	checked
U02	The client accepts a composite host signature only when both the ML-DSA-44 and the Ed25519 verification succeed under the composite key type it negotiated, so a reply that carries only one valid component, or that names another key or signature type, is rejected.	`network`	proved	agree	checked
X01	The exchange hash H of a connection the client accepted, which is also the session identifier, is not derivable by an attacker who has only the network; it is bound to the shared secret K.	`network`	proved	secret_h	checked
X02	No two accepted sessions share a session identifier: each accept corresponds to a distinct server run and to the client's own fresh key exchange values, so a message bound to one session is not valid in another.	`network`	proved	inj	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-20261114T224652, 2026-11-14. Status: finished. Model: deepseek-v4-1-flash-261017. Steps: 38, prover 10 s, wall 12 min, cost 0.099 USD.

ProVerif: 50 proofs run, 50 decided, 0 without an answer, 6 s, 18 answers recorded.

What was analysed

Protocol	SSH composite host authentication
Version	draft-ietf-sshm-composite-sigs-00
Analysed	The Ed25519+ML-DSA composite host signature and the MLKEM768X25519 KEX from the fixed draft; both components are verified, with the full exchange hash, the roles and the algorithm negotiation bound; trusted known-hosts provisioning; user authentication and rekey are excluded; algorithm identifiers are taken as in the actual draft, with no invented values.
Not analysed	The symbolic model does not prove the computational or post-quantum security of the algorithms; attack traces hold only under explicit assumptions and are not directly equivalent to specification vulnerabilities. Every property must have a reachability/positive-example control.
Source text	228 numbered passages (37431 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	Algorithm specifications
main	4.1	Key generation
main	4.1.1	ssh-mldsa44-ed25519
main	4.1.1.1	Composite raw public key
main	4.1.1.2	Composite raw private key
main	4.1.2	ssh-mldsa87-p384
main	4.1.2.1	Composite raw public key
main	4.1.2.2	Composite raw private key
main	4.1.3	SSH Public Key Format
main	4.1.4	SSH Private Key Format
main	4.2	Signatures
main	4.2.1	Signature contexts
main	4.2.2	Signature Generation
main	4.2.2.1	ssh-mldsa44-ed25519
main	4.2.2.2	ssh-mldsa87-p384
main	4.2.2.3	SSH Signature Format

Document	Section	Title
main	4.2.2.4	Signature Verification
main	4.2.2.4.1	ssh-mldsa44-ed25519
main	4.2.2.4.2	ssh-mldsa87-p384
main	5	Security Considerations
rfc10042	2	PQ/T Hybrid Key Exchange
rfc10042	2.1	PQ/T Hybrid Key Exchange Method Abstraction
rfc10042	2.2	PQ/T Hybrid Key Exchange Message Numbers
rfc10042	2.3	PQ/T Hybrid Key Exchange Method Names
rfc10042	2.3.1	mlkem768nistp256-sha256
rfc10042	2.3.2	mlkem1024nistp384-sha384
rfc10042	2.3.3	mlkem768x25519-sha256
rfc10042	2.4	Shared Secret K
rfc10042	2.5	Key Derivation
rfc4253	7	Key Exchange
rfc4253	7.1	Algorithm Negotiation
rfc4253	7.2	Output from Key Exchange
rfc4253	7.3	Taking Keys Into Use
rfc4253	8	Diffie-Hellman Key Exchange
rfc4253	8.1	diffie-hellman-group1-sha1
rfc4253	8.2	diffie-hellman-group14-sha1
rfc4253	9	Key Re-Exchange

The purpose of the protocol as the source step located it in the documents: The Ed25519+ML-DSA composite host signature and the MLKEM768X25519 KEX from the fixed draft; both components are verified, with the full exchange hash, the roles and the algorithm negotiation bound; trusted known-hosts provisioning; user authentication and rekey are excluded; algorithm identifiers are taken as in the actual draft, with no invented values.

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	agreement on exchange hash and shared secret	the client C is assured that the host key K_S, the exchange hash H with all its fields (V_C, V_S, I_C, I_S, C_INIT, S_REPLY) and the shared secret K it accepted are those of one run of the server S	RFC 4253 says the exchange hash authenticates the key exchange and lists the fields it covers; RFC 10042 gives the hybrid variant of H	s0217, s0218, s0144, s0205

ID	Name	About	What the documents say	Where
A02	server authentication to the client	the client C is assured that a protocol instance of the named server S ran, i.e. that it is talking to the holder of the composite host key pinned for S	RFC 4253 says the key exchange uses explicit server authentication and that the client verifies the host key against a local database; RFC 4251 says the transport performs server host authentication	rfc4253:1428, s0154, s0209, s0205
A08	negotiated algorithms bound in the signature	the client C is assured that the exchange hash under its accepted host signature covers the algorithm name-lists I_C and I_S it exchanged and the host key algorithm it selected, and that verification uses that selected algorithm	RFC 10042 lists I_C and I_S among the fields of the hybrid exchange hash; RFC 4253 says the signature algorithm is negotiated with KEXINIT and that the exchange hash covers both KEXINIT payloads	s0143, s0144, s0169, s0157
A04	distinct server run per accepted session	every accept of host authentication by the client C corresponds to a different run of the server S, so a recorded reply cannot be replayed into a new session	RFC 4253 says the exchange hash of the first key exchange is the unique session identifier and is not changed by later re-exchanges; RFC 4251 says public key authentication exchanges are bound to the session and cannot be replayed in other sessions	rfc4251:849, s0187, s0161
K01	secrecy of host keys and session keys	the composite host private key (mldsa_seed, ed25519_seed) is kept from everyone but the server S, and the session keys derived from K and H are kept from the network attacker	RFC 4253 derives all encryption and integrity keys from K and H; RFC 4251 asks for forward secrecy of session keys and warns that collecting traffic now and decrypting it later must not work	rfc4251:1021, s0189, s0197, s0138, s0015
C08	composite forgery needs both components	the client C is assured that a composite host signature accepted for the server S was made with both component private keys; an adversary who can forge one component cannot authenticate as S	main says an adversary must forge both ML-DSA and Elliptic Curve signatures to forge a composite signature, and that the composite stays EUF-CMA while at least one component is unbroken	main:447, main:455, s0096, s0098, s0088
K04	keys distinct per purpose and direction	the server S and the client C use component key material only inside this composite key, and the two directions of the connection use different derived keys	main forbids reusing component key material between composite and non-composite keys or between composite keys; RFC 4253 derives the two directions of keys from different letters	main:451, s0097, s0190, s0191
D01	no man in the middle host substitution	the client C, holding the composite host key of S from known-hosts, is assured that no attacker on the network replaced the host key or the exchange hash in the exchange	RFC 4253 requires the client to verify that the host key really is the key of S; RFC 4251 states that with a verified association a man in the middle cannot succeed because he cannot sign the session-specific data	rfc4251:905, s0209, s0219
U01	one broken component does not break host auth	the client C is assured that leaking or breaking a single component private key of the server S does not let an attacker authenticate as S	main states the dual-algorithm goal: the composite is at least as secure as the unbroken component, and an adversary must forge both components	main:447, main:469, s0096, s0101, s0083
A08.10	host key bound to the session	the client C is assured that the host key K_S it accepted is bound to this session's exchange hash and session identifier, and not taken from another session	RFC 10042 puts K_S into the hybrid exchange hash; RFC 4253 says the exchange hash is the session identifier used by higher layers	s0144, s0187, s0209
U02	both signature components must verify	the client C is assured that an accepted composite host signature passed both the ML-DSA-44 and the Ed25519 verification under the negotiated algorithm, so neither substituting the signature type nor the key type nor dropping a component is accepted	the composite signature specification requires both component verifications to succeed and the failure of either to fail the composite, and gives the wire format for signature and key type names	s0083, s0088, s0077, s0032

ID	Name	About	What the documents say	Where
X01	secrecy of the exchange hash	the exchange hash H (also the session identifier) of a connection, kept from the network attacker	RFC 4253 says the exchange hash SHOULD be kept secret; RFC 4251 in contrast says the session identifier may be made public without harming security, so the two documents pull apart and the secrecy reading is asked here.	rfc4253:218, rfc4251:849
X02	distinct session identifiers per session	the session identifier of every connection, and the client and server that compute it	RFC 4253 says the exchange hash of the first key exchange is a unique identifier for the connection; RFC 4251 says that if two sessions have the same session id, packets from one can be replayed against the other.	rfc4251:857, main:187

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

The security statements of the documents, paragraph by paragraph

The host listed the 133 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: 32. Not taken: 101. 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
P001	main:444-445	No goal stated	a pointer to the security considerations of the composite signature specification; the goals it states are entered below for rfc10073 section 9
P002	main:447-449	C08, U01, `broken_component`	
P003	main:451-453	K04	
P004	main:455-458	C08, U01, `broken_component`	
P005 to P006	main:462-466	No goal stated	page header and footer
P007	main:469-474	C08, U01, `broken_component`	
P008	main:476-478	The prover cannot be asked	SUF-CMA is a computational statement about producing a second distinct signature for the same message; the symbolic signature model used here has deterministic signatures and cannot express malleability
P009	main:480-482	Outside the scope	non-separability matters when a component signature is checked under a standalone trusted key, which the key-reuse prohibition of P003 rules out; the analysed protocol always verifies both components under the composite key from known-hosts
P010	rfc10042:392-394	K01, A03	
P011	rfc10042:396-406	A04	
P012	rfc10042:408-423	The prover cannot be asked	encoding-length side channels are outside a Dolev-Yao model
P013	rfc10042:425-438	K01	
P014	rfc10042:440-449	No goal stated	advice on which elliptic curve is easier to implement

Paragraphs	Where	Taken up as	Why not
P015	rfc10073:2083-2084	No goal stated	a pointer to the security considerations of ML-DSA
P016 to P017	rfc10073:2088-2109	C08, U01	
P018	rfc10073:2111-2126	No goal stated	migration flexibility and codebase reuse: a deployment argument with nothing that must hold
P019	rfc10073:2130-2142	K04	
P020	rfc10073:2144-2148	C08, U01, `broken_component`	
P021	rfc10073:2152-2156	C08	
P022	rfc10073:2158-2164	C08, U01	
P023	rfc10073:2166-2167	C08, U01, `broken_component`	
P024	rfc10073:2169-2170	C08, U01	
P025	rfc10073:2172-2173	C08, U01, `broken_component`	
P026	rfc10073:2175-2180	Outside the scope	about composite signatures inside X.509 certificates, not the SSH host authentication analysed here
P027 to P031	rfc10073:2184-2212	The prover cannot be asked	strong unforgeability is a computational malleability statement; the symbolic model treats a signature as a deterministic term and cannot produce a second valid signature for the same message
P032 to P035	rfc10073:2216-2248	Outside the scope	separability is about lifting one component signature out of a composite and verifying it under a standalone trusted key; the analysed protocol never trusts a standalone component key (key reuse is forbidden, P003)
P036 to P037	rfc10073:2252-2273	K04	
P038	rfc10073:2275-2286	Outside the scope	certificate revocation checking by a CA
P039	rfc10073:2288-2294	Outside the scope	about application context strings of other protocols; SSH fixes the context to the empty string
P040	rfc10073:2298-2305	Outside the scope	an optional guard inside a standalone traditional verifier; the SSH client verifies the composite type name and the label, not a bare traditional signature
P041 to P042	rfc10073:2309-2326	Outside the scope	policy for deprecating algorithms, not a property of a run
P043	rfc4251:648-650	No goal stated	says only where the security considerations were collected
P044	rfc4251:652-655	A02, A03, K01	
P045	rfc4251:657-662	Outside the scope	user authentication is excluded by the task
P046	rfc4251:664-669	Outside the scope	the connection protocol is excluded by the task
P047	rfc4251:673-684	A04	

Paragraphs	Where	Taken up as	Why not
P048	rfc4251:690-694	The prover cannot be asked	the quality of random data is an assumption of the model, not something the prover can be asked
P049	rfc4251:698-701	Outside the scope	terminal control characters, not cryptography
P050 to P068	rfc4251:707-818	Outside the scope	choice of bulk cipher and CBC/IV handling in the encrypted data channel; the task models no application data, and the model has no encryption of the data channel
P069 to P071	rfc4251:822-835	Outside the scope	integrity of the encrypted data channel, the none-MAC and sequence-number wrap with rekeying; rekey is excluded by the task
P072 to P073	rfc4251:839-855	A04	
P074	rfc4251:857-862	X02	
P075	rfc4251:864-887	Outside the scope	sequence numbers and rekeying of the data channel
P076	rfc4251:891-896	D01, A02	
P077	rfc4251:902-903	D01	
P078	rfc4251:905-932	A02, D01	
P079	rfc4251:934-949	Outside the scope	distribution of host public keys without a trusted association; the task fixes a trusted known-hosts entry as an assumption
P080	rfc4251:955-962	No goal stated	advice on publishing fingerprints
P081	rfc4251:964-982	Outside the scope	manipulation of packets after the session is established, i.e. the data channel
P082	rfc4251:984-989	D01	
P083 to P085	rfc4251:993-1009	The prover cannot be asked	denial of service and CPU cost are outside a trace-based prover
P086	rfc4251:1013-1017	The prover cannot be asked	covert channels are outside a symbolic trace model
P087 to P088	rfc4251:1021-1042	K01, `host_key_after_session`	
P089	rfc4251:1046-1051	A08	
P090	rfc4251:1063-1071	The prover cannot be asked	traffic analysis depends on message lengths and timing, which the model abstracts
P091 to P093	rfc4251:1075-1090	Outside the scope	user authentication and the connection protocol are excluded by the task
P094 to P096	rfc4251:1094-1118	Outside the scope	about running user authentication over a weak transport
P097	rfc4251:1122-1139	Outside the scope	debug messages of the upper layers
P098 to P103	rfc4251:1143-1182	Outside the scope	local security policy of the server (rhosts, agent forwarding and similar), not the key exchange
P104 to P106	rfc4251:1186-1201	Outside the scope	password and host-based user authentication are excluded by the task
P107 to P108	rfc4251:1219-1232	Outside the scope	security of the endpoints themselves

Paragraphs	Where	Taken up as	Why not
P109 to P115	rfc4251:1236-1287	Outside the scope	proxy forwarding and X11 forwarding, i.e. parts of the system the task leaves out
P116	rfc4253:1428-1431	A02, A03, K01	
P117	rfc4253:1433-1434	No goal stated	a pointer to the architectural document
P118	rfc5656:708-712	K01	
P119 to P128	rfc5656:714-795	Outside the scope	choice and strength of elliptic curves and hash functions: computational parameters of the primitives, which the model assumes ideal, and the NIST curves are not the curve of the analysed method
P129	rfc5656:797-803	The prover cannot be asked	quality of the random number generator
P130	rfc8731:165-166	No goal stated	says only that other security considerations are inherited
P131 to P132	rfc8731:168-185	The prover cannot be asked	strength estimates of curve sizes, and the mpint encoding side channel
P133	rfc8731:187-190	No goal stated	a recommendation of which curve to prefer

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: 3. Without: 20. 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 C	party	Everything lost by definition	every target here states what the client is assured of; with the client controlled by the attacker there is no client left to assure, and the only values it holds are its own session keys and the public host key
S02	Server S	party	`host_key_after_session`	
S03	Attacker	party	Nothing gained	the adversary of the model is the party that already has the network
S04	Host public key K_S	value	Nothing gained	the host public key is public and is sent in the key exchange
S05	exchange hash H	value	Everything lost by definition	the exchange hash is the value whose secrecy is asked; holding it loses that goal and reveals no key material
S06	Shared secret K	value	Everything lost by definition	all session keys are derived from K; an attacker holding K has that session's traffic by definition
S07	Session identifier	value	Nothing gained	the session identifier may be made public without harming security (RFC 4251), and the identification strings, name-lists, role strings and the cookie are sent in the clear or are public constants
S08	Server identity	value	Nothing gained	the session identifier may be made public without harming security (RFC 4251), and the identification strings, name-lists, role strings and the cookie are sent in the clear or are public constants

ID	Party or value	Kind	Attacker	Why none
S09	Negotiated algorithm name-lists I_C/I_S	value	Nothing gained	the session identifier may be made public without harming security (RFC 4251), and the identification strings, name-lists, role strings and the cookie are sent in the clear or are public constants
S10	Role identifiers V_C/V_S	value	Nothing gained	the session identifier may be made public without harming security (RFC 4251), and the identification strings, name-lists, role strings and the cookie are sent in the clear or are public constants
S11	Session identifier/exchange hash	value	Nothing gained	the session identifier may be made public without harming security (RFC 4251), and the identification strings, name-lists, role strings and the cookie are sent in the clear or are public constants
S12	cookie	value	Nothing gained	the session identifier may be made public without harming security (RFC 4251), and the identification strings, name-lists, role strings and the cookie are sent in the clear or are public constants
S13	Composite host private key	value	`host_key_after_session`	
S14	Session encryption key	value	Everything lost by definition	holding a session's encryption or integrity key gives that session's traffic by definition; what survives is asked under forward secrecy
S15	Integrity key	value	Everything lost by definition	holding a session's encryption or integrity key gives that session's traffic by definition; what survives is asked under forward secrecy
S16	Composite host signature	value	Nothing gained	the composite signature is sent on the wire in the key exchange reply
S17	exchange hash	value	Everything lost by definition	the same value as S05: its secrecy is the goal itself
S18	Component private key	value	`host_key_after_session`	
S19	Direction keys	value	Everything lost by definition	both direction keys are derived from the same K and H, so an attacker holding a derived key of a session has that session; the target only asks that the two directions be distinct values
S20	Host public key	value	Nothing gained	the host public key is public
S21	Composite signature	value	Nothing gained	the composite signature is sent on the wire
S22	Negotiated algorithm names	value	Nothing gained	the negotiated algorithm names and the signature type name are sent in the clear in KEXINIT and in the reply
S23	Signature format	value	Nothing gained	the negotiated algorithm names and the signature type name are sent in the clear in KEXINIT and in the reply

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		
`broken_component`	the ability to produce valid ML-DSA-44 component signatures for any message under the server's composite host public key without the component private key (one signature algorithm broken); the same capability for the classical component is the symmetric case	main says an adversary must forge both components, and that the composite is EUF-CMA if at least one component is; the underlying specification says the guarantee degrades when one component is broken	main:447, main:469, s0096, s0101, s0098
`host_key_after_session`	the composite host private key (both component private keys) read from the server's storage after the connection has ended	RFC 4251 asks for forward secrecy of session keys and warns about harvest-now-decrypt-later collection; a stored host key is the long-term value that may leak later	rfc4251:1021, main:96, s0187, s0197

ID	`network`	`broken_component`	`host_key_after_session`
A03	Holds		
A02	Holds		
A08	Holds		
A04	Holds		
K01	Holds		Holds
C08		Holds	
K04	Holds		
D01	Holds		
U01		Holds	
A08.10	Holds		
U02	Holds		
X01	Holds		
X02	Holds		

Source and targets

Source: 228 units, 37431 bytes, frozen in source-delivery-01.json. Targets: 10 approved (A03, A02, A08, A04, K01, C08, K04, D01, U01, U02); 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-8bf82a0f2d0f8709318672c88f08

This step as it reports itself: 1 requests (1 failed), 0.000 USD, 31993 tokens in and 4895 out, 31 s.

Notes on the prover and the model

- A03: answered in ProVerif, and the analyst names dh_algebra for this target
- A04: answered in ProVerif, and the analyst names dh_algebra for this target

- K01: answered in ProVerif, and the analyst names dh_algebra for this target
- X01: answered in ProVerif, and the analyst names dh_algebra for this target
- X02: answered in ProVerif, and the analyst names dh_algebra for this target

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

What the analyst named for targets:

step	targets	features	reason
29	A03, A04, K01, X01, X02	dh_algebra, many_sessions	These targets rest on the hybrid key exchange: X25519 is modelled with the single Diffie-Hellman law g^xy=g^yx and no other curve algebra, ML-KEM is a free encapsulation function with a matching decapsulation, and the hash is ideal. ProVerif covers any number of sessions, but the algebraic laws of the curve and the encodings are simplified.
29	A02, A08, A08.10, C08, D01, K04, U01, U02	plain_primitives, many_sessions	These targets rest on the composite host signature only: ML-DSA-44 and Ed25519 are ideal signature schemes with matching verification, the composite combiner and its M' prefix are represented by one function, and SHA-512 is ideal. The case in which the ML-DSA component is broken is modelled by an explicit forging function for that component; the classical case is the symmetric reading.

The analyst's closing note

The model covers the plain run of draft-ietf-sshm-composite-sigs-00 with mlkem768x25519-sha256: KEXINIT both ways, SSH_MSG_KEX_HYBRID_INIT, the reply with K_S, S_REPLY and both component signatures, K = HASH(K_PQ || K_CL), the exchange hash over V_C, V_S, I_C, I_S, K_S, C_INIT, S_REPLY and K, the direction keys from HASH(K || H || letter || session_id), and a client that checks both components, the key and signature type names and its known-hosts entry.

All thirteen targets have answers on model.pv, all as proofs with a control that fails: agreement on host key, exchange hash and shared secret (A03, A08, A08.10, D01, U02, and under the broken-component attacker also C08 and U01), injective agreement (A04, X02), secrecy of the host private key and of the session keys (K01, including forward secrecy against the host key read from storage after the connection, and the exchange hash, X01), the no-forgery property when one component signature algorithm is broken (C08, U01), and direction key separation (K04). The control runs falsify the same properties where a client verifies only one component, where a server reads out its own session state, and where both directions use one derivation input, which shows the proofs depend on the checks they name.

No finding: no weakness appeared that the documents do not already state. What blocks more: the model idealises the primitives (a broken component is a forgery capability, X25519 has one Diffie-Hellman law, ML-KEM and the hashes are ideal), so the computational claims and the per-adversary-class degradation cannot be checked here; the SUF-CMA failure and non-separability that the documents discuss need deterministic-malleability or key-reuse reasoning that a symbolic prover cannot express; and the length checks of RFC 10042, rekeying, user authentication and the encrypted data channel are outside the model. K04 is answered only in part: the key-reuse prohibition is a rule about generating keys, not a property of a run.