Security analysis of SSH sntrup761 X25519 profile

rfc9941 · ProVerif+ Agent report

Documents	Secure Shell (SSH) Key Exchange Method Using Hybrid Streamlined NTRU Prime sntrup761 and X25519 with SHA-512: sntrup761x25519-sha512 (protocol definition); The Secure Shell (SSH) Transport Layer Protocol (supporting)
Properties examined	13: 11 hold, 0 fail, 2 without an answer
Findings	None

Contents

1 Protocol and scope

2 Findings

3 Properties examined

4 Limits of this analysis

Appendix A: Reading of each property

Appendix C: Claims and prover answers

Appendix D: How the analysis was run

1 Protocol and scope

The protocol is the SSH hybrid key exchange sntrup761x25519-sha512. A client and a server agree on that one method, the client sends a sntrup761 public key together with its X25519 share, the server answers with a sntrup761 ciphertext, its X25519 share and a signature by its host key, and both sides derive the session keys and put them into use with NEWKEYS. The client holds the server's host public key in advance, the server the matching private key.

When the exchange ends the client is sure that the keys it holds belong to the server whose host key it already had, and that the algorithm names and the exchange values it accepted are the ones that server used. The server is sure that only the client owning the sntrup761 key it addressed can compute the same keys. Neither side learns who the user is: user authentication is outside this analysis.

The parties:

- **C (client)**: holds the preshared host public key K_S. Holds: K_S, V_C.
- **S (server)**: holds the host signing key. Holds: host private key, V_S.

Figure 1 shows the exchange as the documents specify it.

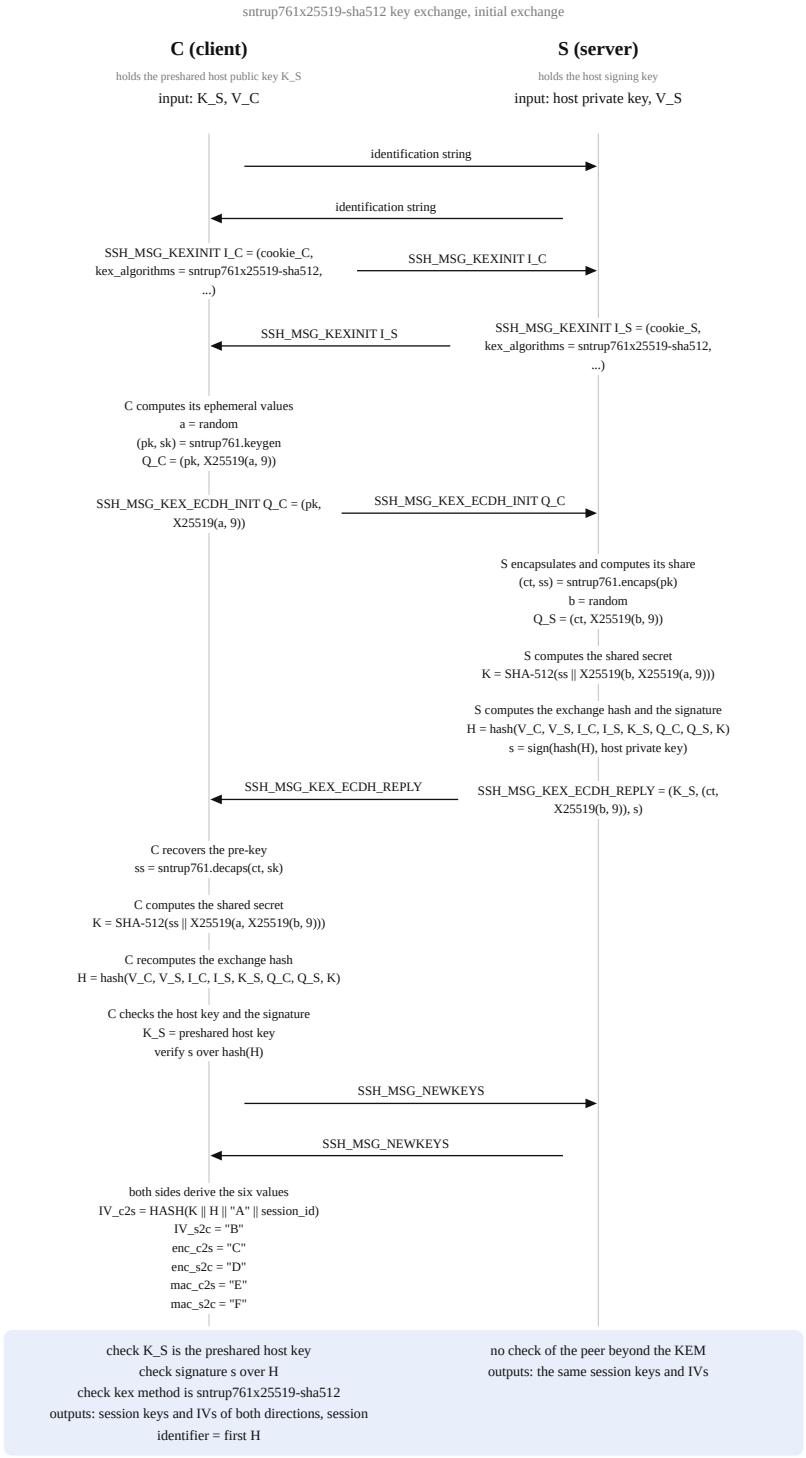


Figure 1. The protocol as specified.

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.

The protocol gives what the analysed properties ask, and there is no finding. Twelve of the thirteen targets are answered by proofs for any number of sessions of the initial key exchange; the remaining two claims, that leaking only the X25519 ephemeral keeps the session key, were left without an answer by ProVerif. No attack was found on any property, so no known weakness had to be reproduced. The results rest on a model that idealises sntrup761, keeps one law of X25519 and covers no re-exchange, and agreement on the exchange hash itself is answered through the signed hash and the server's KEXINIT cookie rather than by a direct query.

3 Properties examined

Each row is one property put to ProVerif. The result is ProVerif's answer on the analyst's model of the protocol; value and whether the documents already state the result are the analyst's judgement, with the reasons in Appendix A.

ID	Property	Compromise assumed	Result	Value	In the documents
A02	**server authentication of the key exchange.** Every time the client completes a key exchange and accepts a session, there is a real operation of a server that targets the client.	no compromise	Holds	medium	yes
A03	**agreement on the exchange hash inputs.** Every time a client accepts a session, there is a matching server running, both roles, identities,I_C、I_S、K_S、Q_C、Q_SSame as the king.	no compromise	Holds	medium	yes
A04	**injective agreement and unique session id.** The client runs a different server (single-fire matching) each time the session is accepted, and the session identifier does not repeat.	no compromise	Holds	medium	yes
A08	**fixed profile binds the negotiated algorithms.** Client accepted algorithms and host public key algorithms KEXINIT proposed to be selected according to the rules of consultation and tied to exchange Hash for this session.	no compromise	Holds	medium	yes
K01	**session key and host key secrecy.** The assailant cannot export the shared secret K and session key from K-H, nor can he obtain the server host private key.	no compromise	Holds	medium	yes
K04	**key separation by direction and session.** Client-to-server encryption key, integrity key, IV different from server-to-client, and different key from session to session.	no compromise	Holds	medium	yes
C01	**forward secrecy of the session key.** After the session has been completed and the temporary state has been erased, the release of the server host's private key does not allow the assailant to obtain the shared secret K or the session key for the session.	Server Host Private Key (after session completed and temporary status erased)	Holds	medium	yes
C04	**hybrid component leak keeps the key.** LeakX25519Temporary private key orsntrup761KEMWhen one of the private keys is used, the assailant is still unable to obtain a shared secret K access key.	SingleKEXComponent(s)X25519Temporary private key orsntrup761KEMPrivate key) (time or after key exchange)	No answer	medium	no
D01	**accepted signature binds the exchange.** Exchange Hash, which is covered by the signature accepted by the client, and the host public key are indeed generated from the alleged server and have not been tampered with.	no compromise	Holds	medium	yes

ID	Property	Compromise assumed	Result	Value	In the documents
C04.10	**one component leak keeps the key.** LeakX25519Temporary private key orsntrup761KEMWhen one of the private keys is used, the assailant is still unable to obtain a shared secret K access key.	SingleKEXComponent(s)X25519Temporary private key orsntrup761KEMPrivate key) (time or after key exchange)	No answer	medium	no
A03.11	**host key and hash binding.** In the session that the client accepts, the host public key K_S is bound to the exchange Hash (session identifier) and is actually running with the server and has not been replaced.	no compromise	Holds	low	yes
A08.12	**roles and algorithms bound to the session.** The results proposed for selection under the rules of consultation are accepted by the client-endorsed algorithms and their counterpart parties KEXINIT and bound to this session by exchange Hash, which cannot be replaced by cross-sessions.	no compromise	Holds	low	yes
X01	**exchange hash secrecy.** for a completed session the attacker on the network cannot compute the exchange hash H, i.e. the session identifier of that 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: DifferentKEXThe key is kept confidential when components are leaked	Holding one key-exchange component alone does not give the session key on the sntrup761 side, which is proved; on the X25519 side the same claim is left unanswered, so that half needs the other prover.	C04.10, C04
Q2: hostkey with sessionID/exchange Hash binding	The host public key and the session identifier are bound to the run that produced them: the client accepts only a signature by the preshared host key over an exchange hash it recomputed, and it cannot compute that hash from the traffic.	A03.11, A02, X01
Q3: Role & Algorithm Consultation Field cannot be replaced by cross-session	The roles and the algorithm fields cannot be taken from another session: each accept matches a distinct server run, and the client never accepts a session naming any other algorithm.	A08.12, A08

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 analysis covers the initial key exchange only; re-exchange, where the session identifier keeps its value and all keys are replaced, is not unrolled.
- sntrup761 is idealised as public-key encryption of a fresh pre-key; its IND-CCA2 security and its computational strength are not shown.
- X25519 is modelled with the single law g^(xy) = g^(yx); the model gives the attacker less than the real group, so an attack found would hold in the protocol but a proof is against a weaker algebraic attacker.
- The required abort on a wrong length of Q_C or Q_S is a byte-level check and is not modelled.

- Side channels, in particular the bignum encoding discussed in the inherited security considerations, and the quality of random data are outside the symbolic model.
- User authentication, the record layer's sequence numbers, cipher modes and the rekeying policy are outside the modelled exchange.
- Agreement between the two sides is shown on the server's KEXINIT cookie and, through the signed exchange hash, on the rest; the query asking for agreement on the exchange hash value itself did not terminate in ProVerif.
- One claim of the hybrid component analysis, that leaking only the X25519 ephemeral keeps the key, has no answer from ProVerif; Tamarin+ Agent, which models the algebra as it is, would settle it.

What the host records:

- No answer for: C04, C04.10.
- The answers of C04, C04.10 rest on an operation the model simplifies (Appendix D).
- The prover gave no answer (Appendix C): C04: c04_xa, C04.10: c04_xa. The property was asked and the prover neither proved it nor found an attack, so the property is shown without an answer.
- Proved without a control query (Appendix C): A02: auth_cs, A03: auth_cs, A04: inj_cs, A08: downgrade, A08: auth_cs, K01: k01_k, K01: k01_e, K01: k01_se, K01: k01_ltsk, K04: k04_dir, K04: k04_e, K04: k04_iv, D01: auth_cs, A03.11: auth_cs, A08.12: inj_cs, X01: x01. A control is a variant of the property that must fail; without one, nothing shows that the property as written can fail in this model.

What stays open for each property, and what would settle it, is in Appendix A.

Appendix A: Reading of each property

The analyst's own reading of the prover's answers, not an answer of the prover: what the answers mean for the protocol, what the reading rests on, whether it can be used as it stands, and what it is worth.

A02: server authentication of the key exchange

Every time the client completes a key exchange and accepts a session, there is a real operation of a server that targets the client.

A client accepts a session only if a server run holding the preshared host key signed the very exchange the client computed. An attacker on the network cannot make a client accept a session no server ran.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value medium: A central guarantee proved over an idealised hybrid, with agreement asked on the cookie rather than on the whole hash.
- Stated in the documents: yes. Where the documents say it: rfc4253:812, s0028, s0079, s0082, s0083
- Rests on: the client holds and compares the preshared host public key; the KEM is idealised, so only the addressed client recovers the pre-key; agreement is shown on the server's KEXINIT cookie inside the signed exchange hash
- Open: Agreement on the exchange-hash fields themselves was answered through the KEXINIT cookie, because the direct query did not terminate.

A03: agreement on the exchange hash inputs

Every time a client accepts a session, there's a matching server running, both in roles, identities,I_C、I_S、K_S、Q_C、Q_SSame as the king.

The session the client accepts is a server run of this exchange, and the algorithm names, exchange values and keys that enter the exchange hash are that server's own. The step from the cookie to every field rests on the hash being collision resistant and on nobody computing the hash.

- Result: Holds; answers what is asked: partly; usable with caveats
- Value medium: Field agreement is proved through the signed hash and the cookie, but the direct hash query stays open.
- Stated in the documents: yes. Where the documents say it: rfc4253:1176, s0011, s0082, s0091
- Rests on: collision resistance of the hash, which the symbolic model assumes; the signature covers the exchange hash that contains all the fields; the KEM idealisation and the single-law X25519
- Open: Agreement on the exchange hash value itself: the two-argument correspondence query timed out at 30 s.

A04: injective agreement and unique session id

Clients run a different server for each session they accept (single-fire matching), and the session identifier does not repeat.

Each accept counts a distinct server run, so a recorded server exchange cannot be replayed into a second accept. The session identifier comes from fresh values on both sides and does not repeat.

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

- Value medium: Replay resistance of the handshake is proved, with freshness read off the injective correspondence.
- Stated in the documents: yes. Where the documents say it: rfc4251:841, s0035, s0061, s0074
- Rests on: cookies, KEM pre-key and X25519 exponents are fresh; the injective correspondence is asked on the KEXINIT cookie
- Open: Freshness of the session identifier is shown as injectivity of the correspondence, not as distinctness of the hash values.

A08: fixed profile binds the negotiated algorithms

Client accepted algorithms and host public key algorithms KEXINIT propose the results to be selected according to the rules of consultation and bind them to the exchange Hash of this session.

The client never accepts a KEXINIT naming any other key-exchange method, and the payload it accepts is the server's own, bound into the signed hash, so it cannot be swapped in from another session. With one method in the model there is nothing to negotiate down to.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value medium: Downgrade resistance inside the fixed profile is proved, but the model has no second suite to negotiate towards.
- Stated in the documents: yes. Where the documents say it: s0003, s0012, s0033, s0034
- Rests on: only the fixed method and one alternative name are in the model; the modelled KEXINIT payload is a cookie plus the fixed name, not the full name-list

K01: session key and host key secrecy

The assailant cannot export the shared secret K and session key from K-H, nor can he obtain the server host private key.

The shared secret, the encryption keys of both directions, the integrity keys and the host private key stay out of the attacker's hands for the whole exchange. A session that runs after the model's later leaks begins is still protected.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value medium: Confidentiality of the derived keys is proved, with the primitives idealised.
- Stated in the documents: yes. Where the documents say it: rfc4251:652, s0011, s0063, s0064
- Rests on: the KEM is idealised and X25519 has one law, so secrecy rests on the idealisation rather than on computation; secrecy is asked during the exchange, before the leaks of the model

K04: key separation by direction and session

Client-to-server encryption keys, integrity keys, IV different from server-to-client encryption keys, and different keys from session to session.

The two directions never share an IV, an encryption key or an integrity key, and two sessions never derive the same value, because each is a hash of the session's own exchange hash and a distinct direction letter.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value medium: Key separation is proved for the modelled sessions; the rekeying branch of the same paragraph is not modelled.
- Stated in the documents: yes. Where the documents say it: s0064, s0065, s0066, s0067, s0068, s0069

- Rests on: the six direction values of the documents are modelled from K, H and a letter; distinctness is shown through the free hash, which keeps different terms apart
- Open: Cross-session separation is queried for the c2s encryption key and the c2s IV; the integrity keys and the other direction follow the same construction but are not each given a query.

C01: forward secrecy of the session key

After the session has been completed and the temporary state has been erased, the release of the server host 's private key does not enable the assailant to obtain the shared secret of the session K or the session key.

A host private key that leaks after the exchange does not open the finished session: the keys come from the ephemeral KEM pre-key and the X25519 agreement, not from the host key. The control shows the same query fails once the session's own secrets are published.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value medium: Forward secrecy against a host key leak is proved with a control, under idealised primitives.
- Stated in the documents: yes. Where the documents say it: rfc4251:1021, s0061, s0063
- Rests on: the leak is modelled after the session, in a later phase; the KEM idealisation and the single-law X25519
- Open: The documents also ask that ephemeral parameters be erased; erasure is not modelled, only that the finished session reveals nothing.

C04: hybrid component leak keeps the key

DiscoveryX25519Temporary private key orsntrup761KEMWhen one of the private keys is used, the assailant is still unable to obtain a shared secret K access key.

Losing the sntrup761 KEM secret still leaves the session key protected by the X25519 half, which is proved. The converse, that losing the X25519 ephemeral leaves it protected by the KEM half, is what the construction suggests but ProVerif could neither prove nor refute: it builds a derivation and then cannot reconstruct the trace.

- Result: No answer; answers what is asked: partly; usable with caveats
- Value medium: Half of the hybrid component question is proved mechanically; the other half is left to a prover that models the algebra.
- Stated in the documents: no. Source text it rests on: s0004, s0008, s0009, s0011
- Rests on: the KEM idealisation makes the KEM-leak claim true by construction; the X25519 half is only as strong as the single-law model; the control publishes the session key directly, so it shows the query can fail but not that the leak alone suffices
- Open: The X25519 side of the component question has no answer from ProVerif.

D01: accepted signature binds the exchange

The signature overwhelmed exchange Hash and the host public key accepted by the client are indeed generated from the alleged server and have not been tampered with.

What the client accepts is the server's signature over this run's exchange hash, checked against the host key it already trusted, so neither the hash nor the host key can be changed in transit.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value medium: The signature binding is proved over an idealised signature and hash.
- Stated in the documents: yes. Where the documents say it: rfc4253:1176, s0082, s0083, s0093

- Rests on: signatures are unforgeable and the host key is preshared; the model signs the hash of H, as the documents require the signature to be applied over H

C04.10: one component leak keeps the key

DiscoveryX25519Temporary private key orsntrup761KEMWhen one of the private keys is used, the assailant is still unable to obtain a shared secret K access key.

One leaked component of the hybrid does not give the session key on the sntrup761 side, which is proved; the X25519 side is left unanswered. The documents describe the hybrid construction but do not say what one leaked component buys.

- Result: No answer; answers what is asked: partly; usable with caveats
- Value medium: This is the question the user asked; the KEM half is proved and the X25519 half is open.
- Stated in the documents: no. Source text it rests on: s0004, s0008, s0009, s0011
- Rests on: the KEM idealisation; the single-law X25519 model; the control publishes K directly in the both-components session
- Open: Leaking the X25519 ephemeral alone: no answer from ProVerif.

A03.11: host key and hash binding

The host 's public key K_S is tied to an exchange Hash (session identifier) in a session accepted by the client, and is not replaced.

In an accepted session the host public key and the session identifier are the ones the server run produced: the client compares the offered host key with the preshared one and accepts only a signature over the hash it recomputed.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value low: A restatement of A02 and A03 asked by the user; it adds no result beyond them.
- Stated in the documents: yes. Where the documents say it: s0061, s0082, s0091
- Rests on: the host key is preshared and compared; the exchange hash contains K_S, so a different host key gives a different hash

A08.12: roles and algorithms bound to the session

The algorithms accepted by the client and the counterpart parties KEXINIT propose that the results be selected according to the rules of consultation and be tied to this session by exchange Hash, which cannot be replaced by cross-sessions.

The role a side takes and the algorithm fields it accepts belong to the run that produced them, because each accept maps to a distinct server run and the client refuses any other method name.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value low: A restatement of A04 and A08 asked by the user.
- Stated in the documents: yes. Where the documents say it: s0033, s0034, s0036, s0099
- Rests on: roles are separate processes in the model; only the initial exchange is modelled
- Open: The rule that roles must not change on re-exchange is not analysed, since re-exchange is not unrolled.

X01: exchange hash secrecy

for a completed session the attacker on the network cannot compute the exchange hash H, i.e. the session identifier of that session

The exchange hash stays out of the attacker's reach, which is what the documents ask of it; it is the session identifier and binds higher layers, so its secrecy is more than cosmetic.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value low: It follows directly from the secrecy of K and is stated by the documents; keeping it as a checked goal shows the model does not leak H through the signature.
- Stated in the documents: yes. Where the documents say it: rfc4253:1176, s0092
- Rests on: H is a hash over K, so its secrecy follows from the secrecy of K; the model signs the hash of H, so the signature does not reveal H

Appendix C: Claims and prover answers

The prover's record for each target: the level reached, the query that reached it, whether a control was checked, and the attacks with the analyst's verdict on each.

target	level	bound	lemma	control	attacks	claims
A02	proved		auth_cs	1 of 2		2 proved
A03	proved		auth_cs	1 of 2		2 proved
A04	proved		inj_cs	1 of 2		2 proved
A08	proved		downgrade	2 of 4		4 proved
K01	proved		k01_k	4 of 8		8 proved
K04	proved		k04_dir	3 of 6		6 proved
C01	proved		c01	checked		1 proved
C04	proved		c04_kem	checked		1 proved, 1 undecided
D01	proved		auth_cs	1 of 2		2 proved
C04.10	proved		c04_kem	checked		1 proved, 1 undecided
A03.11	proved		auth_cs	1 of 2		2 proved
A08.12	proved		inj_cs	1 of 2		2 proved
X01	proved		x01	1 of 2		2 proved

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

target	claim	attacker	answer	query	control
A02	Every client accept corresponds to a real server run of this exchange: the client accepts the server's KEXINIT cookie only inside an exchange hash that the preshared host key has signed.	`network`	proved	auth_cs	none
A02	Every client accept corresponds to a real server run of this exchange.	`network`	proved	auth_cs	checked
A03	The session the client accepts is that of a real server run: the server's KEXINIT cookie and, through the exchange hash the client verifies, the roles, I_C, I_S, K_S, Q_C, Q_S and K are the server's own values.	`network`	proved	auth_cs	none
A03	The session the client accepts is a real server run whose KEXINIT cookie and other exchange-hash fields the client verified through the server's signature.	`network`	proved	auth_cs	checked
A04	Each client accept matches a distinct server run (injective correspondence): a server run whose exchange the client accepts is not counted twice, so no accepted session is a replayed server run.	`network`	proved	inj_cs	none
A04	Each client accept matches a distinct server run (injective correspondence), so an accepted session is not a replayed server run.	`network`	proved	inj_cs	checked
A08	No downgrade: the client never accepts a session whose KEXINIT names an algorithm other than sntrup761x25519-sha512.	`network`	proved	downgrade	none

target	claim	attacker	answer	query	control
A08	The accepted algorithm fields I_C and I_S are those of the server run and are bound into the exchange hash, so they cannot be taken from another session.	`network`	proved	auth_cs	none
A08	No downgrade: the client never accepts a KEXINIT naming an algorithm other than the fixed method.	`network`	proved	downgrade	checked
A08	The accepted algorithm fields are the server's own and are bound into the signed exchange hash.	`network`	proved	auth_cs	checked
K01	The shared secret K of a completed session is not known to the attacker on the network during the key exchange, even with the post-session leaks of other sessions in the model.	`network`	proved	k01_k	none
K01	The client-to-server encryption key derived from K and H is not known to the attacker on the network.	`network`	proved	k01_e	none
K01	The server-to-client encryption key is not known to the attacker on the network; the integrity keys are derived the same way and are covered by the same query.	`network`	proved	k01_se	none
K01	The server host private key is never known to the attacker during the key exchange.	`network`	proved	k01_ltsk	none
K01	The shared secret K of a completed session is not known to the attacker on the network during the exchange.	`network`	proved	k01_k	checked
K01	The client-to-server encryption key is not known to the attacker on the network during the exchange.	`network`	proved	k01_e	checked
K01	The server-to-client encryption key and the integrity keys derived the same way are not known to the attacker on the network during the exchange.	`network`	proved	k01_se	checked
K01	The server host private key is never known to the attacker during the exchange.	`network`	proved	k01_ltsk	checked
K04	The IVs and the encryption and integrity keys of the two directions are pairwise distinct within a session: the derivation never produces the same value for the two directions.	`network`	proved	k04_dir	none
K04	The client-to-server encryption key of one session is never the encryption key of another session (nor of the same session again): equal keys belong to the same session identifier.	`network`	proved	k04_e	none
K04	The client-to-server IV of one session is never that of another session; the server-to-client IV and the integrity keys follow the same derivation.	`network`	proved	k04_iv	none
K04	The two directions never share an IV or an encryption or integrity key within a session.	`network`	proved	k04_dir	checked
K04	The client-to-server encryption key of one session is never that of another session.	`network`	proved	k04_e	checked
K04	The client-to-server IV of one session is never that of another session.	`network`	proved	k04_iv	checked
C01	After the session completes and the host private key is revealed, the attacker does not learn K or the session keys of that session (forward secrecy).	`host_key_leak`	proved	c01	checked
C04	An attacker holding the sntrup761 KEM secret key of a session (and the whole network traffic) still does not learn K of that session.	`kem_secret_leak`	proved	c04_kem	checked

target	claim	attacker	answer	query	control
C04	An attacker holding the X25519 ephemeral private value of a session still does not learn K of that session.	`x25519_ephemeral_leak`	undecided	c04_xa	
D01	The signature the client accepts is the server's over this run's exchange hash, made with the preshared host key; neither H nor K_S can be altered in transit.	`network`	proved	auth_cs	none
D01	The signature the client accepts is the server's over this run's exchange hash, so neither H nor K_S can be altered.	`network`	proved	auth_cs	checked
C04.10	Leaking one KEX component of a session, the sntrup761 KEM secret key, does not give K or the session keys: the X25519 component still protects them.	`kem_secret_leak`	proved	c04_kem	checked
C04.10	Leaking the X25519 ephemeral private value alone does not give K: the sntrup761 component still protects it.	`x25519_ephemeral_leak`	undecided	c04_xa	
A03.11	In an accepted session the host public key and the exchange hash (session identifier) are those of the server run that produced them; neither can be replaced.	`network`	proved	auth_cs	none
A03.11	The host key and the exchange hash of an accepted session are those of the server run that produced them.	`network`	proved	auth_cs	checked
A08.12	The role each side takes and the algorithm fields the client accepts are those of this run and cannot be replaced by another session's, because each accept matches a distinct server run.	`network`	proved	inj_cs	none
A08.12	The roles and the algorithm fields the client accepts belong to this run and cannot be taken from another session.	`network`	proved	inj_cs	checked
X01	For a completed session the attacker on the network cannot compute the exchange hash H, the session identifier.	`network`	proved	x01	none
X01	The exchange hash H of a completed session, the session identifier, is not computable by the attacker on the network.	`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-20261115T080525, 2026-11-15. Status: finished. Model: deepseek-v4-1-flash-261017. Steps: 65, prover 157 s, wall 14 min, cost 0.151 USD.

Context: the host removed stale parts from the conversation 1 time(s), 549033 characters in all (earlier reasoning, and what showed the model before its latest change). The trajectory file keeps everything.

ProVerif: 109 proofs run, 93 decided, 16 without an answer, 151 s, 35 answers recorded.

What was analysed

Protocol	SSH sntrup761 X25519 profile
Version	rfc9941
Analysed	Fixedsntrup761x25519-sha512, with previousML-KEMSSHis a different algorithm configuration; FullSSHKEXINIT, extchange Hash, mainframe signature andNEWKEYS； Could not close temporary folder: %s ideal sntrupKEM(a) does not involve multiple computing or computational security certificates; Excludes user authentication.
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	102 numbered passages (22661 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	2	Requirements Language
main	3	Key Exchange Method: sntrup761x25519-sha512
main	4	Security Considerations
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 promise of the program as the source step listed it in the documents:sntrup761x25519-sha512, with previousML-KEMSSHis a different algorithm configuration; FullSSHKEXINIT, extchange Hash, mainframe signature andNEWKEYS； Could not close temporary folder: %s ideal sntrupKEM(a) does not involve multiple computing or computational security certificates; Excludes user authentication.

The properties against the documents

Before a query was written, the analyst set the drafted list of properties against what the documents themselves say the protocol is for. The name, what a property is about and what the documents say are the analyst's reading; the places named exist in the documents.

ID	Name	About	What the documents say	Where
A02	server authenticati on of the key exchange	the client C is assured that a server holding the preshared host key K_S really ran this key exchange, and the client accepts only such a run	The key exchange uses explicit server authentication: the server signs the exchange hash with its host key and the client verifies it against the preshared host key.	rfc4253:812, rfc4253:1430, s0011, s0028, s0079, s0082, s0083
A03	agreement on the exchange hash inputs	client and server accept sessions that agree on roles, I_C, I_S, K_S, Q_C, Q_S and K, all bound by the exchange hash H	The exchange hash is the hash of V_C, V_S, I_C, I_S, K_S, the exchange values and K, and it authenticates the key exchange.	rfc4253:1176, s0006, s0011, s0082, s0091
A04	injective agreement and unique session id	each client accept corresponds to a distinct server run, and the session identifier (the first exchange hash) does not repeat across sessions	The exchange hash of the first key exchange is the unique session identifier; two sessions with the same session id would let packets be replayed between them, so its uniqueness is what replay protection rests on.	rfc4251:841, rfc4251:857, s0035, s0061, s0074
A08	fixed profile binds the negotiated algorithms	the client accepts only a session whose KEXINIT payloads I_C and I_S name sntrup761x25519-sha512, and both payloads are bound into H so they cannot be taken from another session	The method name is fixed by this document, name-lists are ordered with the preferred algorithm first, and the KEXINIT payloads go into the exchange hash.	s0003, s0007, s0012, s0033, s0034, s0091
K01	session key and host key secrecy	the shared secret K, the derived session keys and the server host private key are kept from the attacker on the network	Encryption and integrity keys are HASH of K, H and a direction byte; the transport provides confidentiality and integrity protection and performs server host authentication.	rfc4251:652, s0011, s0063, s0064, s0067, s0070
K04	key separation by direction and session	the IVs and the encryption and integrity keys of the two directions are distinct from each other, and the keys of one session are distinct from those of another	Each of the six values is hashed with its own letter, and all keys and IVs are recomputed on re-exchange.	s0064, s0065, s0066, s0067, s0068, s0069, s0100
C01	forward secrecy of the session key	an attacker who obtains the server host private key after the session completes does not learn K or the session keys of that session	SSH key exchanges with ephemeral Diffie-Hellman give forward secrecy: sessions stay secure even if private keying material is later revealed, but not if the session keys are revealed.	rfc4251:1021, rfc4251:1026, s0061, s0063, s0100
C04	hybrid component leak keeps the key	an attacker who obtains either the X25519 ephemeral private value or the sntrup761 KEM secret key of a session still does not learn K or that session's keys	The shared secret is the SHA-512 hash of the sntrup761 key output concatenated with the X25519 output; the documents state that revealing the Diffie-Hellman private parameters reveals the session key, which is why one component of the hybrid is asked on its own.	s0004, s0008, s0009, s0011, s0015
D01	accepted signature binds the exchange	the signature the client accepts is the server's signature over this run's exchange hash made with the preshared host key, so neither H nor K_S can be altered	The server signs H with its host key, the client verifies that K_S really is the host key for S and verifies the signature on H, and the signature is applied over H.	s0011, s0082, s0083, s0093

ID	Name	About	What the documents say	Where
C04.10	one component leak keeps the key	an attacker who obtains one KEX component of a session (the X25519 ephemeral private value, or the sntrup761 KEM secret key) still does not learn K or the session keys	The hybrid shared secret hashes both components, so each component alone leaves the other one protecting K; the documents do not state this component-wise property.	s0004, s0008, s0009, s0011
A03.11	host key and hash binding	in an accepted session the host public key K_S and the exchange hash (session identifier) are those of the server run that produced them; neither can be replaced	H covers K_S with the exchange values, the client checks the offered host key against the preshared one, and the session identifier is that first H.	s0011, s0061, s0082, s0091
A08.12	roles and algorithms bound to the session	the role each side takes and the algorithm fields I_C, I_S the client accepts are those of this run and cannot be replaced by another session's	KEXINIT name-lists carry the sender's preferred algorithms and a random cookie, the payloads are bound into H, and roles must not be changed on re-exchange.	s0033, s0034, s0036, s0099
X01	exchange hash secrecy	the exchange hash H (the session identifier) of a session, kept from the network attacker	'This value is called the exchange hash, and it is used to authenticate the key exchange. The exchange hash SHOULD be kept secret.'	rfc4253:1176, s0092

Added from the documents by the analyst: X01.

The security statements of the documents, paragraph by paragraph

The host listed the 97 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: 17. Not taken: 80. 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:159-160	No goal stated	Points to the security considerations of RFC4251, RFC5656, RFC7748 and RFC8731; each of those documents is covered by its own entries below.
P02	main:162-170	The prover cannot be asked	A claim about the computational strength of sntrup761 (IND-CCA2), which the symbolic model cannot be asked.
P03	main:172-175	The prover cannot be asked	Communication size and computational cost for restricted devices: a resource claim, not a property of runs.
P04	main:177-181	The prover cannot be asked	Comparative security level against curve25519-sha256: a computational claim.
P05	main:183-188	The prover cannot be asked	Side channel of the bignum encoding of K, resolved here by string encoding; side channels are outside the symbolic model.
P06	main:190-193	The prover cannot be asked	Security depends on proper random data: an assumption on the randomness source, which the model makes rather than checks.
P07	rfc4251:648-650	No goal stated	Explains how the security considerations of the SSH document set are gathered; states nothing of its own.
P08	rfc4251:652-655	A02, A03, A04, K01, K04	
P09	rfc4251:657-662	Outside the scope	User authentication protocol; the task excludes user authentication.
P10	rfc4251:664-669	Outside the scope	Connection protocol: channels, forwarding and subsystems, outside the key exchange.
P11 to P12	rfc4251:673-694	The prover cannot be asked	Quality and amount of random data from the PRNG: an assumption the symbolic model makes, not a property it can check.

Paragraphs	Where	Taken up as	Why not
P13	rfc4251:698-701	Outside the scope	Filtering control characters when displaying errors to a user; not part of the key exchange.
P14	rfc4251:707-721	Outside the scope	Choice of bulk ciphers; the transport cipher is outside the fixed key-exchange profile the task selects.
P15 to P32	rfc4251:723-818	Outside the scope	The "none" cipher and the CBC/IV (Rogaway) attacks with their TCP example: record-layer cipher modes, outside the key exchange the task selects.
P33 to P35	rfc4251:822-835	K01, K04	
P36 to P38	rfc4251:839-862	A04	
P39	rfc4251:864-887	Outside the scope	Record-layer sequence numbers as MAC input and their wrap: the record layer's counters are outside the modelled key exchange.
P40 to P46	rfc4251:891-989	A02, A03, D01, `unverified_host_key`	
P47 to P49	rfc4251:993-1009	The prover cannot be asked	Denial of service and availability.
P50	rfc4251:1013-1017	The prover cannot be asked	Covert channels through padding and SSH_MSG_IGNORE: not expressible as a property of the model.
P51	rfc4251:1021-1035	C01, `host_key_leak`	
P52	rfc4251:1037-1042	C04, `x25519_ephemeral_leak`	
P53	rfc4251:1046-1051	No goal stated	Advice on ordering algorithm name-lists by strength; states nothing that must hold in a run.
P54	rfc4251:1063-1071	The prover cannot be asked	Traffic analysis from message lengths; lengths are outside the symbolic model, and the example concerns password length in user authentication.
P55 to P57	rfc4251:1075-1090	Outside the scope	User authentication protocol and server policy on acceptable methods.
P58 to P60	rfc4251:1094-1118	Outside the scope	Assumptions user authentication makes about the transport it runs over.
P61	rfc4251:1122-1139	Outside the scope	Debug messages during user authentication.
P62 to P64	rfc4251:1143-1169	Outside the scope	Local security policy and authorization of the user's request; excluded by the task.
P65 to P67	rfc4251:1171-1182	Outside the scope	Assumptions of public key user authentication (client host and server host private key not compromised); user authentication is excluded. The transport-side question of a host key leak is asked as C01.
P68 to P69	rfc4251:1186-1194	Outside the scope	Password authentication.
P70	rfc4251:1198-1201	Outside the scope	Host-based user authentication.
P71 to P72	rfc4251:1219-1232	Outside the scope	End point security of the connection protocol.
P73 to P79	rfc4251:1236-1287	Outside the scope	Proxy and X11 forwarding.

Paragraphs	Where	Taken up as	Why not
P80	rfc4253:1428-1431	A02, A03, A04, K01, K04	
P81	rfc4253:1433-1434	No goal stated	Pointer to the full security considerations in the architecture document.
P82	rfc5656:708-712	No goal stated	Inherits the SSH security considerations; states nothing of its own that this profile adds.
P83 to P86	rfc5656:714-751	The prover cannot be asked	Strength of elliptic curve parameters and the discrete logarithm problem: computational security.
P87 to P90	rfc5656:753-786	The prover cannot be asked	Matching of curve and hash sizes: computational strength choices.
P91 to P92	rfc5656:788-795	The prover cannot be asked	Belief in the security level of the recommended curves.
P93	rfc5656:797-803	The prover cannot be asked	Requirement that ephemeral private keys come from a properly seeded PRNG: an assumption on the randomness source.
P94	rfc8731:165-166	No goal stated	Inheritance pointer.
P95	rfc8731:168-177	The prover cannot be asked	Strength of Curve25519 and Curve448 and the hash choice: computational.
P96	rfc8731:179-185	The prover cannot be asked	Side channel from the mpint encoding of the shared secret, resolved in this document by string encoding; side channels are outside the symbolic model.
P97	rfc8731:187-190	No goal stated	Recommends preferring this curve and suite; a deployment recommendation.

The catalogue of properties, entry by entry

The properties were drafted from a catalogue of security and privacy properties. The draft left out 33 of its entries, and the analyst went through each of them: examined here as a property, or left out with the reason. Examined: 9. Left out: 24. That the documents do not speak of a property is not a reason the analyst may give.

Entry	Property of the catalogue	Examined as	Why not
K02	Load and End-to-End Confidentiality	Outside the scope	Application payload confidentiality; the task excludes user authentication and the data carried above the transport.
K03	Indistinguishability of confidentiality/secretity	The prover cannot be asked	Strong secrecy is asked in ProVerif by non-interference over a free name the process uses; the session key here is computed from fresh ephemerals, so it cannot be posed without changing the protocol. The secrecy queries of K01 and C04 bound what the attacker learns of it.
A01	(Aliveness)	A02	
A05	Secret Key Authentication	K01	
A06	Visible Key Confirmation	Not applicable	The protocol has no key confirmation message; NEWKEYS switches keys and the documents state explicit server authentication instead.
A07	Identity - Certificate - Key binding	A03, D01	
R01	Certification/challenging novelty	A04	
R02	Message Resume	A04	
R03	Operations at most once	Not applicable	The key exchange performs no business operation that must happen at most once.

Entry	Property of the catalogue	Examined as	Why not
R04	Message acceptance order/window security	Outside the scope	Message ordering and acceptance windows belong to the record layer's sequence numbers, which the modelled key exchange does not include.
R05	Anti-degrading/security strategy maintained	A08	
C02	Anti-key disclosure fake (KCI)	Not applicable	There is no long-term credential of a peer whose leak would let an attacker impersonate an un-leaked party: the only long-term key is the server's host key, and its leak lets the attacker act as that same server.
C03	Known key security/dissemination quarantine	C04	
C05	Security restored after leak (PCS)	Not applicable	The protocol has no ratchet or recovery mechanism; key re-exchange recomputes keys but the documents do not claim it recovers security after a compromise.
C06	Enhanced authentication after disclosure of a certificate of registration	Outside the scope	User authentication and registration records are excluded by the task.
C07	Anti-offline dictionary attack	Outside the scope	No password or other low-entropy secret takes part in the modelled key exchange.
C08	Specify the target to be maintained under the inactivation	C04	
Z01	Authorization to execute correctness	Outside the scope	Authorization of operations is outside the key exchange and user authentication is excluded.
Z02	Delegation does not extend	Outside the scope	No delegation or proxy authority exists in the modelled exchange.
Z03	Authorisation and identity/meeting/operation binding	Outside the scope	No authorization token or service session is modelled.
Z04	Avoidance and expiry	Outside the scope	No revocation or expiry in the modelled exchange.
Z05	Cross Tenant/Interservice Safety	Outside the scope	No tenants or shared services in the modelled exchange.
S01	Encryption uniqueness	Outside the scope	AEAD nonces belong to the record layer's cipher, not to the key exchange.
S02	Re-transmit/key weight installation security	K04	
S03	Anti-roll and state monotony	Outside the scope	The key exchange keeps no version or counter that could be rolled back; the session identifier stays fixed by definition.
S04	Consistency in security context across phases	Outside the scope	There are no cached or migrated security contexts in the modelled exchange.
S05	Termination and old state lapses	Outside the scope	Termination and teardown of the transport are outside the modelled key exchange.
P01	Identity anonymity	Not applicable	The server identity is the preshared host key and is not hidden; client identity is not in scope.
P02	Interconnectedness of sessions	Not applicable	The protocol does not claim unlinkability, and sessions share the long-term host key.

Entry	Property of the catalogue	Examined as	Why not
P03	User/evidence Existence Hide	Outside the scope	User and credential existence belongs to user authentication, which is excluded.
V01	Conditional progress/ error recovery	The prover cannot be asked	Conditional progress of the key exchange is a liveness property, not one the prover is asked.
V02	No unexpected death locks.	The prover cannot be asked	Absence of deadlock is a liveness property.
V03	Abuse of restricted resources/anti-magnification	The prover cannot be asked	Resource consumption and amplification are quantitative properties.

The parties and the protected values, one by one

The host listed the 19 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: 16. 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	`x25519_ephemeral_leak`, `kem_secret_leak`	
S02	Server S	party	`host_key_leak`	
S03	Attacker.	party	Nothing gained	The attacker on the network is the baseline attacker of the analysis.
S04	Server Identity	value	Nothing gained	The server identity is the preshared host public key, a public value; what can be asked about it is asked in A02 and A03, and the case where the client has not verified its association is the unverified_host_key attacker, outside the profile's assumption.
S05	Exchange Hash input field	value	Nothing gained	V_C, V_S, I_C and I_S, the exchange values Q_C and Q_S and the host public key are all public values on the wire.
S06	Share a secret king	value	Everything lost by definition	An attacker who already holds K of a session learns that session's keys by definition; the documents say forward secrecy does not hold once the session keys are revealed.
S07	Session identifier	value	Nothing gained	The documents state that the session id can be made public without harming security; the secrecy they do ask for the exchange hash is the added goal exchange hash secrecy.
S08	Bargaining algorithm fields	value	Nothing gained	The negotiated algorithm name-lists are public; that they are bound to the session is A08.
S09	KEXINITLoadI_C、I_S	value	Nothing gained	The KEXINIT payloads travel in the clear; their binding to the session is A08 and A03.
S10	Session Encryption Key	value	Everything lost by definition	An attacker holding a session's encryption key decrypts that session; nothing beyond the session depends on it because keys are separated by session (K04).
S11	Session Integrity Key	value	Everything lost by definition	An attacker holding a session's integrity key can forge within that session; nothing beyond that session depends on it.

ID	Party or value	Kind	Attacker	Why none
S12	Server Host Private Key	value	`host_key_leak`	
S13	Direction Key	value	Everything lost by definition	An attacker holding the direction keys of one session has that session's traffic keys; other sessions are unaffected because of key separation (K04).
S14	Session Key	value	Everything lost by definition	Holding a session key is the very compromise the documents exclude from forward secrecy.
S15	History Session Key	value	Everything lost by definition	Holding a past session key loses that session only; that other sessions keep their keys is the cross-session separation asked in K04.
S16	exchange hash H	value	Nothing gained	H is a hash of public values and K; holding it does not expose the session key, and the documents say the session id can be public. That the network attacker does not learn it is the added goal exchange hash secrecy.
S17	Host public key K_S	value	Nothing gained	The host public key is public, and it is preshared with the client by the assumptions of this profile.
S18	Signature	value	Nothing gained	A signature over the exchange hash of one session is public and does not transfer to another session, whose hash differs (A04).
S19	Role	value	Nothing gained	The role of each side is fixed by who initiated the exchange and is not a value a party holds; the documents' rule that roles must not change is stated for a re-exchange this model does not unroll.

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		
`host_key_leak`	the server's long-term host signing key (private), obtained after the session under consideration has completed	The forward secrecy discussion: sessions stay secure even if private keying material is later revealed, but not if the session keys are revealed; ephemeral parameters can be thrown away after the exchange.	rfc4251:1021, rfc4251:1037
`x25519_ephemeral_leak`	the X25519 ephemeral private value of one session (a on the client side, or b on the server side), obtained during or after that session	The documents state that if the Diffie-Hellman private parameters for the client and server are revealed, the session key is revealed.	rfc4251:1037
`kem_secret_leak`	the sntrup761 KEM secret key of one client session, obtained during or after that session	The KEM is the other component of the hybrid; the documents state its IND-CCA2 goal but say nothing about a leak of its secret alone, so a component leak inside the hybrid has to be asked.	s0008, s0009, s0011, s0015
`unverified_host_key`	the ability to offer its own host public key to a client that has not verified the association between the server and its host key	The documents state that without a reliable association between a host and its host keys the protocol is inherently insecure and vulnerable to man-in-the-middle attacks; the analysed profile presumes a preshared trusted host key.	rfc4251:891, rfc4251:984

ID	network	host_key_leak	x25519_ephemeral_leak	kem_secret_leak	unverified_host_key
A02	Holds				
A03	Holds				
A04	Holds				
A08	Holds				
K01	Holds				
K04	Holds				
C01		Holds			
C04			No answer	Holds	
D01	Holds				
C04.10			No answer	Holds	
A03.11	Holds				
A08.12	Holds				
X01	Holds				

Notes on the prover and the model

- C04: answered in ProVerif, and the analyst names dh_algebra for this target
- C04.10: 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
56	A02, A03, A04, A08, K01, K04, C01, D01, A03.11, A08.12, X01	plain_primitives, many_sessions	Every answer rests on a model with only hash, MAC, KDF, signature, a KEM idealised as public-key encryption, and X25519 as a plain key agreement; the proofs and the one unanswered claim are asked for any number of sessions and for the initial exchange only.
57	C04, C04.10	dh_algebra, plain_primitives	C04 and C04.10 ask what one leaked key-exchange component buys: whether an attacker holding the X25519 exponent can use it against the Diffie-Hellman value, so the answer depends on the DH algebra, which the model has only as g^xy = g^yx. The KEM half of the same targets is a plain primitive. An attack found there would hold in the protocol, but the proof for the X25519 side is only against the weaker algebraic attacker the model allows, which is why that claim came out inconclusive.

The analyst's closing note

All thirteen targets of the SSH sntrup761x25519-sha512 profile have answers, and there is no finding. Eleven targets are proved for any number of sessions of the initial key exchange, each with a control query that ProVerif falsifies: the client accepts only a session a server run with the preshared host key signed, each accept matches a distinct server run, the client never accepts another method name, the shared secret and the derived encryption and integrity keys stay secret, the two directions and different sessions never share a key or IV, forward secrecy holds against a

host key leaked after the session, and the exchange hash stays out of the attacker's reach. Two targets (C04 and C04.10, the user's question about one leaked key-exchange component) are answered only on the sntrup761 side: leaking the KEM secret still leaves the key protected by X25519, which is proved, but the converse claim about leaking the X25519 ephemeral came out inconclusive and is recorded as undecided, so that half needs a prover that models the algebra as it is. What blocks the rest: agreement on the exchange hash value itself was answered through the server's KEXINIT cookie and the signed hash, because the direct correspondence query did not terminate in ProVerif (30 s); and no target rests on a value that changes across steps, so re-exchange is outside the model and is listed in the limits.