Security analysis of WEBBOTAUTH: signed requests, key discovery and identity binding

This report in another language: 中文.

draft-ietf-webbotauth-httpsig-protocol-01 · ProVerif+ Agent report

Documents	HTTP Message Signatures for automated traffic (protocol definition); HTTP Message Signatures (supporting)
Properties examined	10: 6 hold, 4 fail (5 attacks)
Findings	None; known results reproduced in Appendix B: 3

Contents

1 Protocol and scope
2 Findings
3 Properties examined
4 Limits of this analysis
Appendix A: Reading of each property
Appendix B: Known results reproduced
Appendix C: Claims and prover answers
Appendix D: How the analysis was run

1 Protocol and scope

The protocol lets an automated client, the Agent, have its HTTP requests recognised by an Origin. The Agent signs each request with a key it controls and names the https URL where its key set is published in the Signature-Agent header; the Origin resolves that URL to a JWK Set and verifies the signature, so the identifier it ends up with is that URL together with a key the URL provides.

When the exchange ends the Origin is sure that a holder of a key published at the resolved URL signed the components the signature covers. It is sure of nothing else: the protocol says nothing about who operates the Agent or whether the request is authorised, and the request can still be replayed inside the signature lifetime.

The parties:

- **Agent**: signs each request with its own signing key. Holds: signing private key, Signature-Agent URL.
- **Origin**: verifies the request and attributes it. Holds: its authority (@authority).
- **Directory**: serves the JWK Set at the Signature-Agent URL. Holds: published public keys.

Figure 1 shows the exchange as the documents specify it.

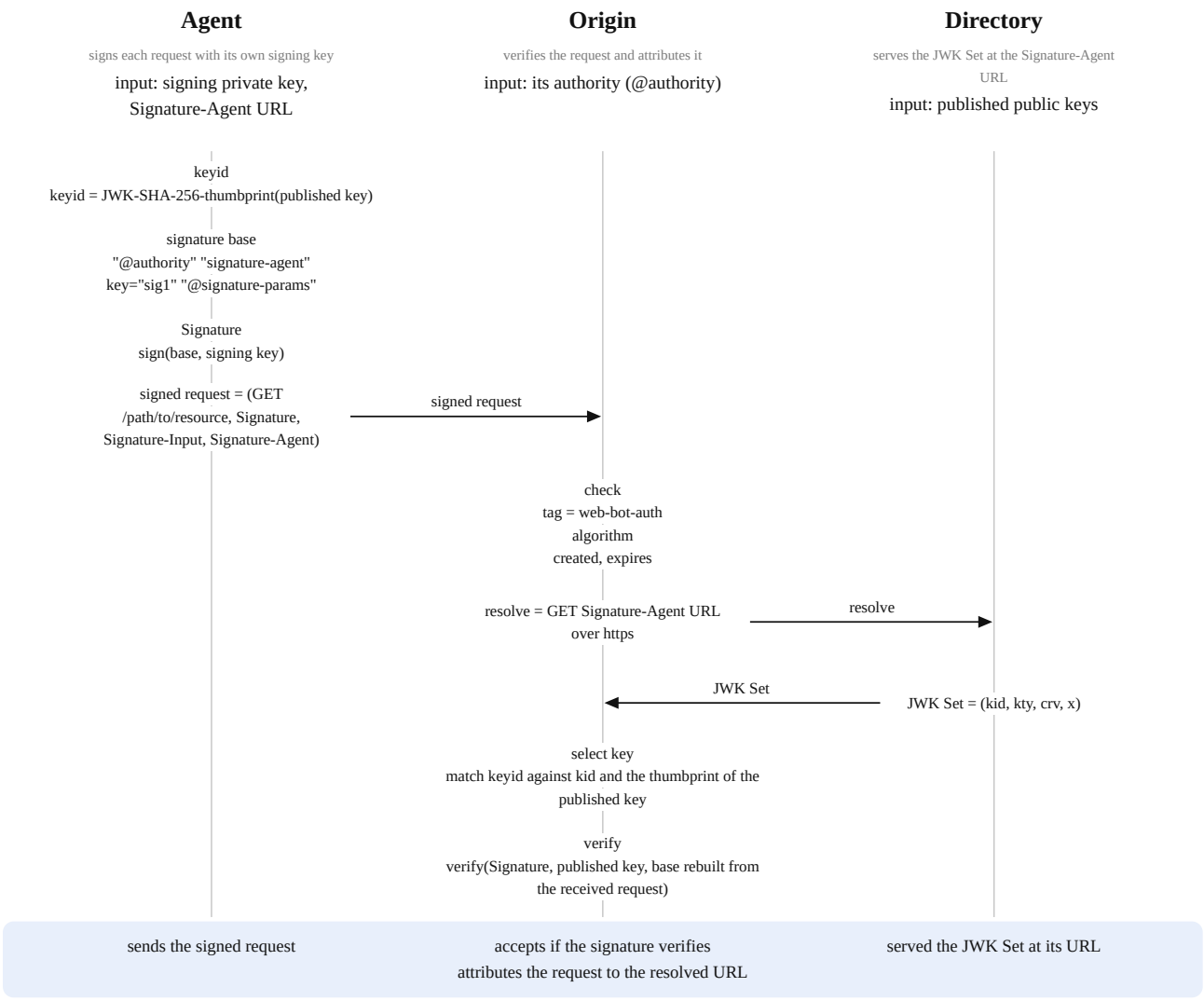


Figure 1. The protocol as specified.

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. Where a property fails, the documents know it: 3 such results are reproduced in Appendix B.

The protocol gives what it is meant to give inside the scope analysed: every accepted request is attributed to a Signature-Agent URL that published the verifying key, and the covered components are exactly the ones the Agent signed. No finding: each property that fails is one the documents already state, and the model reproduces it with its exact conditions. A recorded signature is accepted again unchanged until it expires, and with the minimal compliant coverage set it is also accepted for a different method, path and body. An exfiltrated signing key lets the adversary be attributed to the Agent's URL until verifiers refresh their directory, and the identifier an Agent carries lets origins correlate it across sites. Two remedies were tried on variants of the model: covering method, path and digest removes the substitution, while adding a nonce and a table of accepted nonces did not remove the replay, because the lookup and the record are two steps. All of this rests on an ideal signature, an idealised TLS-authenticated fetch, an honest directory, and no model of the cache or the expires window; the durations are not established.

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
D01	**signature attributed to resolved URL.** Whenever the Origin accepts a request with a web-bot-auth signature, there is a real signing event by an Agent whose signing key belongs to the Signature-Agent URL the Origin resolved for that request (or the fallback keyid fingerprint), and the covered components match the request's actual values.	no compromise	Holds	medium	yes
A03	**identifier matches a real Agent run.** Whenever the Origin accepts a signed request, there is a real Agent run whose published Signature-Agent URL (or fallback keyid fingerprint) matches the identifier the Origin attributes it to, and the covered components match the values the Agent actually signed.	no compromise	Holds	medium	yes
A07	**URL binds the key it publishes.** The Signature-Agent URL the Origin attributes a request to must be the URL that actually published the verifying key; there is no acceptance event that attributes a request signed with a key of URL_A to URL_B.	no compromise	Holds	medium	yes
R02	**one acceptance per signature instance.** Within the signature's expires lifetime, the same signed message instance is accepted by the same Origin at most once; the fewer components are covered, the more request dimensions can be substituted, but substitution must not go beyond what the signature covers.	no compromise	Attack holds (stated in the documents)	medium	yes
K01	**signing key leak boundary.** While the signing private key is not leaked, the attacker cannot obtain it; once the private key leaks, the attacker can impersonate the Agent until the directory is refreshed and the signature expires. This boundary should be stated explicitly rather than treated as confidentiality holding.	Agent's signing private key (leak after the session/request completes, to delimit the consequences of a leak; no leak in the non-leak case)	Attack holds (stated in the documents)	medium	yes
R05	**pinned algorithm and key use.** The algorithms and keys of signatures the Origin accepts satisfy this draft's policy: asymmetric signatures that are not shared HMAC, keys not reused across purposes, and algorithm and key material consistent; an attacker cannot lower the accepted configuration below that baseline by substituting the algorithm or the key.	no compromise	Holds	low	yes
A07.7	**key lookup keyed on URL.** The Signature-Agent URL that a signature accepted by the Origin is attributed to must be the URL that actually published the verifying key, and the key lookup is keyed on the (URL, key) pair.	no compromise	Holds	medium	yes
D01.8	**covered components match the request.** In requests the Origin accepts, the components the signature covers (@authority, signature-agent and the selected @method/@path/@target-uri/Content-Digest) match the values the Agent actually signed and were not substituted.	no compromise	Holds	medium	yes

ID	Property	Compromise assumed	Result	Value	In the documents
U01	**coverage sets and replay boundary.** Within the signature's validity period, coverage set (a) allows the same signature to be replayed on requests with a different method/path/body under the same authority; coverage set (b) makes any substitution of method/target-uri/body fail verification, and the same signed message is accepted at most once.	no compromise	Attack holds (stated in the documents)	medium	yes
X01	**requests from one Agent are unlinkable.** No two Origins ever attribute requests to the same (Signature-Agent URL, keyid) identifier; an Agent's requests cannot be correlated across origins by the identifier they carry.	no compromise	Attack holds (stated in the documents)	low	yes

"Attack holds (stated in the documents)": the prover found the attack, and the documents themselves already state it. The property fails; this is a known result, not a finding (Appendix B).

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: the accepted signature is bound to the URL/public key pair the key was obtained from	Yes, and the model proves it: an accepted signature is attributed to a URL that published exactly the verifying key, because the lookup is on the (URL, key) pair and the key set is fetched from that URL, so a key learned from another URL cannot be attributed here.	A07.7, A07
Q2: origin and integrity of requests for the selected covered fields	For the components the signature covers, the origin of the request and its integrity hold: the Origin rebuilds the signature base from the request it received, so a covered value that was changed fails verification. Components the signature does not cover can be changed freely, which is the minimal coverage set's known boundary.	D01.8, U01
Q3: replay and request-substitution boundary of the minimal coverage set versus the method/target-uri/body-digest coverage set	With the minimal coverage set a recorded signature is accepted for a different method, path and body on the same authority; with the stronger set any such replacement fails. Neither set stops the identical request from being accepted again, so the replay boundary is the signature lifetime, not the coverage set.	U01, R02

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 signature is an ideal one-way function with no algebraic structure; collision, key-substitution-by-transformation and non-deterministic-primitive issues are out of the model.
- The TLS-authenticated fetch of the Signature-Agent URL is modelled as a private channel: the model takes the document's requirement that the JWK Set comes from the host named in the URL as satisfied and does not model TLS itself.
- The directory is an honest publisher of the key for its URL; a malicious operator is treated as the holder of that identifier.

- The directory cache, its lifetime, key rotation and removal, and the created/expires window are not modelled. The replay and key-compromise results show that the situation is reachable, not how long it lasts, so no bound on the window is established.
- The model cannot express an atomic test-and-record of a nonce, so the nonce remedy could not be confirmed; a real implementation needs a single locked update.
- Redistribution of key material and the directory response signature of Appendix B (Section 5.5.3) are not modelled at all; the model only knows keys the Origin fetched itself.
- Serialisation and verification of the signature base are RFC 9421's and not modelled: the model works on typed values, so field parsing, canonicalisation and HTTP version ambiguities are outside it.
- Signature labels and multiple signatures on one request are not modelled; the model verifies one signature per request against the member it covers.
- Unsigned requests, Accept-Signature, sessions and status codes are not modelled, and availability is outside what a trace prover can show.

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 the prover's answer: 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.

D01: signature attributed to resolved URL

Whenever the Origin accepts a request carrying a web-bot-auth signature, there is a real signing event by an Agent whose signing key belongs to the Signature-Agent URL the Origin resolved for that request (or the fallback keyid thumbprint), and the covered components match the request's actual values.

Every request the Origin accepts is attributed to a Signature-Agent URL that published the key which verifies it, and that key's holder really signed the covered components. The only way to be accepted without a real Agent run is to hold the Agent's private key.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value medium: The documents state the guarantee (Main 4.4) and the model establishes it under the idealisations listed, which is a proof of a central property rather than a new result.
- Stated in the documents: yes. Where the documents say it: s0006, s0008, s0021, s0074, s0111
- Rests on: the signature is an ideal one-way function; the JWK Set is fetched from the host named in the URL (TLS taken as satisfied); the directory is honest for its own URL; the cache and the expires window are not modelled; one signature per request
- Next step: Model the directory cache and key removal to bound how long a stale association keeps verifying.

A03: identifier matches a real Agent run

Whenever the Origin accepts a signed request, there is a real run of an Agent whose published Signature-Agent URL (or fallback keyid thumbprint) matches the identifier the Origin attributes it to, and the covered components match the values the Agent actually signed.

The identifier an accepted request is attributed to is one under which an Agent really ran, either the URL that published the key or the keyid thumbprint used on fallback. No acceptance creates an identifier out of nothing.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value medium: It confirms the identification rule Main 4.1 states, with the same idealisations as D01.
- Stated in the documents: yes. Where the documents say it: s0006, s0007, s0008, s0020
- Rests on: the signature is ideal; the fetch is TLS-authenticated; the fallback to the keyid thumbprint is modelled only through the key published at the URL
- Next step: Add the fallback path in which the Origin verifies with a key held out of band and attributes to the thumbprint only.

A07: URL binds the key it publishes

The Signature-Agent URL the Origin attributes to must be the URL that actually published the verifying key; there is no acceptance event that attributes a request signed with URL_A's key to URL_B.

An accepted request is attributed to the URL that published the verifying key, never to another URL, so a party cannot have its identifier attached to a key it did not publish, and a key from one URL cannot carry a request to another.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value medium: Main 5.4 states the (URL, key) requirement; the proof shows the model meets it, and the control shows a verifier that takes the key set from anywhere breaks it, which is worth having because such a verifier would fail silently.
- Stated in the documents: yes. Where the documents say it: s0006, s0011, s0111, s0122
- Rests on: the JWK Set is fetched from the host named in the URL; the directory is honest for its own URL; redistributed key material is not modelled
- Next step: Model a verifier that indexes by keyid alone to exhibit the failure Main 5.4 warns about; the model's lax-verifier control shows the property would break there.

R02: one acceptance per signature instance

While the signature's expires is valid, the same signed message instance is accepted by the same Origin at most once; the fewer covered components there are, the more request dimensions can be substituted, but never beyond what the signature covers.

A recorded signature is accepted again unchanged by the next Origin session inside its lifetime; the strong coverage set does not prevent that because the replayed message is identical. What a substitution can change never goes beyond the covered components, so the authority and the Signature-Agent URL always stay as signed.

- Result: Attack holds (stated in the documents); answers what is asked: yes; usable
- Value medium: Main 5.2.3 and 6.9 state both halves; the model shows the replay with the exact message, confirms the strong set does not help, and shows that a nonce check written as two steps does not close it either, which is the precise condition an implementer needs.
- Stated in the documents: yes. Where the documents say it: s0051, s0076, s0188, s0189
- Rests on: one signature instance per Agent run; no directory cache or expires arithmetic; no nonce parameter in the main model; the nonce variant keeps the replay because the check and the record are two steps
- Open: how long the window is: the cache lifetime and expires are not modelled; whether a nonce cache removes the replay when the test and the record are atomic
- Next step: The nonce remedy needs a model with an atomic test-and-record; without it the replay cannot be shown to be removed.

K01: signing key leak boundary

While the signing private key is not leaked, the attacker cannot obtain it; once the private key leaks, the attacker can impersonate the Agent until the directory refresh and the signature expiry, and this boundary should be made explicit rather than assumed to hold by secrecy.

Without exfiltration the signing key stays with the Agent: the attacker only learns it if it is revealed. With the key the attacker is accepted and attributed to the Agent's URL although the Agent never ran, and the protocol has no way to cut that short beyond the signature lifetime and the verifier's next directory refresh.

- Result: Attack holds (stated in the documents); answers what is asked: yes; usable with caveats
- Value medium: RFC 9421 7.3.2 and Main 6.3 state exactly this boundary; the model shows it mechanically, with the request that gets through.

- Stated in the documents: yes. Where the documents say it: s0049, s0139, s0159, s0160
- Rests on: the key is revealed outright, with no bound on when; the directory is not refreshed within the trace; expires is not modelled
- Open: how long a removed key keeps verifying: the cache is not modelled
- Next step: Model key removal and the cache lifetime to state the duration of the window.

R05: pinned algorithm and key use

The algorithm and key used by a signature the Origin accepts satisfy this draft's policy: asymmetric signatures, not shared HMAC; the key is not reused across purposes; the algorithm and the key material are consistent; and an attacker cannot make the accepted configuration fall below that baseline by substituting an algorithm or a key.

No accepted signature uses an algorithm other than the pinned asymmetric one and the verifying key is always one the attributed URL publishes. A shared-secret HMAC, or public material used as a shared key, cannot buy acceptance from a verifier that keeps the algorithm pinned.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value low: Main 6.4 forbids HMAC outright; the proof shows a compliant verifier cannot be talked into it, but a model whose compliant verifier contains no symmetric path cannot say much more. The lax-verifier control shows the downgrade does go through where the algorithm is taken from the message.
- Stated in the documents: yes. Where the documents say it: s0162, s0164, s0047
- Rests on: the algorithm is pinned in the model, as the task's scope says the protocol fixes it; the compliant verifier has no HMAC path at all, so the negative result is close to a tautology; key reuse policies are not modelled
- Open: key reuse across purposes is a SHOULD that the model does not represent
- Next step: Model a verifier that accepts HMAC keyed with the published public key to show the downgrade RFC 9421 7.3.6 warns about.

A07.7: key lookup keyed on URL

The Signature-Agent URL an accepted signature is attributed to must be the URL that actually published the verifying key, and the key lookup is keyed on the (URL, key) pair.

The URL an accepted signature is attributed to is the URL that published that very key, because the Origin looks the key up on the (URL, key) pair and not on the keyid alone. A key learned under another URL is never used for the asserted one.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value medium: This answers the user's first question and Main 5.4's MUST; the model keeps the (URL, key) association, so the failure Main 5.4 describes is excluded by construction and reappears in the control where the key set is taken from anyone.
- Stated in the documents: yes. Where the documents say it: s0111, s0006, s0022, s0122
- Rests on: the fetch comes from the host named in the URL; the keyid equals the thumbprint of the published key; redistribution is not modelled
- Next step: Nothing further on model.pv: the (URL, key) association is the whole check, and the control exhibits its absence.

D01.8: covered components match the request

In a request the Origin accepts, the components the signature covers (@authority, signature-agent and the chosen @method/@path/@target-uri/Content-Digest) match the values the Agent actually signed, and have not been substituted.

With the stronger coverage set every covered component of the accepted request equals the value the Agent signed, including the content digest, so a changed method, path or body fails verification. With the minimal set the authority and the Signature-Agent URL still match what was signed; only the uncovered fields are free.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value medium: Main 5.2 and RFC 9421 3.2 state it; the model shows the two coverage sets behave differently in exactly the way the draft says.
- Stated in the documents: yes. Where the documents say it: rfc9421:3784, s0052, s0060, s0659
- Rests on: the Origin rebuilds the signature base from the received message; the content digest is a perfect hash of the body; serialisation of the base is RFC 9421's and not modelled
- Next step: Nothing further on model.pv: the boundary between covered and uncovered fields is fully exhibited by the two queries.

U01: coverage sets and replay boundary

Within the signature lifetime, coverage set (a) allows the same signature to be replayed to requests with a different method/path/body under the same authority; coverage set (b) makes any substitution of method/target-uri/body cause verification to fail, so the same signed message is accepted at most once.

With the minimal compliant coverage set, a captured signature is accepted for a different method, path and body on the same authority, so the Origin's view of the request can be changed within that scope. With the stronger set any such change fails, but an identical request is still accepted again inside the signature lifetime.

- Result: Attack holds (stated in the documents); answers what is asked: yes; usable
- Value medium: It answers the user's third question with both readings: the strong set removes substitution but not replay, which is the distinction an operator needs when choosing a coverage set. A variant in which all signers use the strong set was checked and the substitution attack is gone there.
- Stated in the documents: yes. Where the documents say it: s0051, s0052, s0053, s0058, s0076
- Rests on: the two coverage sets are the ones Main 5.2 lists; expires and the cache are not modelled, so no duration is stated; no nonce
- Open: how long the captured signature stays usable
- Next step: Model the nonce parameter with an atomic test-and-record to see whether the draft's deferral to RFC 9421 closes the replay too.

X01: requests from one Agent are unlinkable

No two Origins ever attribute requests to the same (Signature-Agent URL, keyid) identifier; an Agent's requests cannot be correlated across origins by the identifier they carry.

Two origins that both see requests signed by one Agent see the same URL and the same keyid, so they can recognise the Agent and correlate its traffic across sites. Rotation does not break the link while the URL stays the same.

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

- Value low: Main 7.1 states it in as many words; the model reproduces the correlation, which is a known design consequence rather than a new weakness.
- Stated in the documents: yes. Where the documents say it: s0141, s0144
- Rests on: two honest origins in the model; the identifier is the URL plus the keyid; privacy is stated as a property of the identifier, not of traffic analysis
- Next step: Nothing in this model: the document declares the protocol unsuitable for agents that must avoid identification.

Appendix B: Known results reproduced

A property that fails as the documents already state, or that they set aside, is not a finding. What follows is of that kind: the analysis reproduces it on the model, which gives the exact conditions.

B.1 A recorded signature is accepted again until it expires

Kind	Weakness of the protocol as specified
Stated in the documents	main:1178
Value	medium
Properties	R02, U01 (section 3)
Source text	s0051, s0076, s0188, s0189
Shown by	ProVerif attack query `replay`

What happens. A signed request that the adversary records is accepted again, unchanged, by the next Origin session inside the signature lifetime. One Agent run produces one signature and the Origin accepts that signature more than once. The stronger coverage set does not change this, because the replayed message is identical.

What the attacker needs. One signed request, observed on the network. No key and no computation: the adversary resends the bytes it recorded.

What the attacker gains. The replay reaches only what the signature covers: the same authority and the same Signature-Agent URL, and, when the signer omitted the optional components, any method, path and body on that authority until expires.

Proposed change. Recording the nonce parameter of RFC 9421 at the Origin removes the replay only if the lookup and the record are one atomic step. A variant of the model that adds a nonce to the signature parameters and a table of accepted nonces was checked and still lets two Origin sessions pass the check before either records the nonce, so that check does not confirm the remedy. An implementation must make the test and the record a single locked update.

Check of the change with ProVerif. The change was put into a copy of the model: The Agent puts a fresh nonce in @signature-params, which every signature covers, and the Origin keeps a table of nonces it has already accepted and refuses a repeated one. ProVerif still finds the attack for `replay`: the change does not remove it. The normal run of the protocol is still found. The other 21 recorded answers are unchanged. Changed model: `records/fixes/0048-fix-nonce.pv`.

B.2 A signature over the authority alone is valid for any request on it

Kind	Weakness of the protocol as specified
Stated in the documents	main:1140
Value	medium
Properties	U01, D01.8 (section 3)
Source text	s0051, s0052, s0188
Shown by	ProVerif attack query `min`

What happens. When a signature covers only @authority and the signature-agent member, the Origin accepts it for a different method, path and body on the same authority. The covered components are checked and match; the rest of the request is free. The draft states this and keeps the extra components optional.

What the attacker needs. One signed request from an Agent that used the minimal compliant coverage set, i.e. that covered only @authority and the signature-agent member.

What the attacker gains. The Origin attributes to the Agent's URL a request the Agent never made, with any method, path and body the adversary chooses, for the lifetime of the captured signature.

Proposed change. Every Agent covering @method, @path and Content-Digest in addition to @authority and the signature-agent member removes the substitution: a variant of the model in which all signers use that coverage set was checked, the attack query is verified there and every other recorded answer is unchanged. The draft already recommends these components but keeps them optional, so this is a signer policy the document deliberately does not mandate.

Check of the change with ProVerif. The change was put into a copy of the model: Every Agent covers @method, @path and Content-Digest in addition to @authority and the signature-agent member, i.e. the optional components of Section 5.2 are used by all signers, so no signature is valid for a different method, path or body. On the changed model ProVerif proves `min`: the attack is no longer possible there. The normal run of the protocol is still found. The other 22 recorded answers are unchanged. This is a first check on the model: it shows that the change removes this attack there, not that it introduces no other problem. Changed model: `records/fixes/0048-fix-strong.pv`.

B.3 A stolen signing key impersonates the Agent until the directory is refreshed

Kind	Weakness of the protocol as specified
Stated in the documents	rfc9421:3605, main:1062
Value	medium
Properties	K01 (section 3)
Source text	s0159, s0160, s0706, s0139
Shown by	ProVerif attack query `bind_ctl`

What happens. With the signing key the adversary signs its own request for the Agent's URL and the Origin accepts and attributes it, although no Agent run produced that signature. This is the boundary the draft describes: no revocation, and removal from the directory takes effect only at the verifier's next refresh.

What the attacker needs. The Agent's signing private key, exfiltrated at any time before the forged request. The published key set needs no change: the verifier keeps accepting the key it already holds.

What the attacker gains. The adversary has requests accepted and attributed to the Agent's URL until every verifier has refreshed its directory and the signatures have expired; the document defines no revocation, so the Agent cannot shorten the window beyond its signature lifetimes.

Proposed change. The draft names the remedy it can offer: short created and expires values, and removing the key from the directory. A protocol change that would help is a revocation or status signal from the directory that verifiers check before accepting a key, which Section 6.3 says the document does not define.

Check of the change with ProVerif. Not checked: The draft defines no revocation mechanism, so there is no change of messages, checks or derivations to put into a model: a revocation or status signal would be a new message the specification does not define.

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
D01	proved		bind	checked		2 proved
A03	proved		bind	checked		2 proved
A07	proved		published	checked		2 proved
R02	proved		bind	checked	replay (holds_in_source)	2 proved, 1 attack
K01	proved		keysec	checked	bind_ctl (holds_in_source)	1 proved, 1 attack
R05	proved		algpin	checked		3 proved
A07.7	proved		published	checked		2 proved
D01.8	proved		strong	checked		2 proved
U01	proved		strong	checked	min (holds_in_source); replay (holds_in_source)	1 proved, 2 attack
X01	proved		sameid	checked	unlink (holds_in_source)	1 proved, 1 attack

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

target	claim	attacker	answer	query	control
D01	Every request the Origin accepts with a web-bot-auth signature was signed by a holder of the key that the resolved Signature-Agent URL publishes (the key is bound to the URL the Origin fetched it from), unless the Agent's signing key has been given to the attacker. The attacker has the network only.	`network`	proved	bind	checked
D01	The covered components of an accepted request (authority, signature-agent member, and with the stronger set @method, @path and the content digest) carry exactly the values the Agent signed. The attacker has the network only.	`network`	proved	strong	checked
A03	The identifier the Origin attributes an accepted request to (the Signature-Agent URL, or the keyid thumbprint where it holds the key by other means) is one under which an Agent really ran and signed, unless the signing key was given to the attacker. The attacker has the network only.	`network`	proved	bind	checked
A03	The covered component values of the accepted request are those the Agent actually signed. The attacker has the network only.	`network`	proved	strong	checked
A07	No acceptance attributes a request signed with a key published at one URL to a different URL. The attacker has the network only.	`network`	proved	bind	checked
A07	The URL an accepted signature is attributed to is a URL that published exactly that verifying key in its JWK Set. The attacker has the network only.	`network`	proved	published	checked
R02	A substituted request never reaches beyond the covered components: whatever the attacker changes, the authority and Signature-Agent URL the Origin attributes are the ones the Agent signed. The attacker has the network only.	`network`	proved	bind	checked

target	claim	attacker	answer	query	control
R02	Within the expires window the same signature instance is accepted at most once by one Origin. The attacker has the network only: it records one request and replays it unchanged.	`network`	attack (holds_in_source)	replay	
R02	With the stronger coverage set, any replacement of method, target URI or body makes verification fail. The attacker has the network only.	`network`	proved	strong	checked
K01	Once the signing key leaks, the attacker has requests accepted and attributed to the Agent's URL with no Agent run at all, until the verifier's directory is refreshed and the signatures expire. The attacker has the Agent's signing private key.	`key_theft`	attack (holds_in_source)	bind_ctl (the control of bind)	
K01	When the signing key is not exfiltrated the attacker never obtains it: knowledge of the private key implies the key was revealed. The attacker has the network only.	`network`	proved	keysec	checked
R05	An accepted signature is made by a holder of the URL's private key, so public key material cannot be used as a shared secret to forge acceptance. The attacker has the network only.	`network`	proved	bind	checked
R05	No accepted signature uses an algorithm other than the pinned asymmetric one, so a symmetric HMAC or a downgraded algorithm keyed with public material cannot be substituted. The attacker has the network only.	`network`	proved	algpin	checked
R05	The key that verifies an accepted signature is one the attributed URL publishes, not an attacker-chosen key (no key specification mixup). The attacker has the network only.	`network`	proved	published	checked
A07.7	An accepted signature attributed to a URL was made by a holder of that URL's key, so no key from another URL is used for the attribution. The attacker has the network only.	`network`	proved	bind	checked
A07.7	The URL the Origin attributes an accepted signature to is the URL that published that very key: the lookup is on the (URL, key) pair, so a key learned from another URL cannot be attributed to the asserted one. The attacker has the network only.	`network`	proved	published	checked
D01.8	With the stronger coverage set, every covered component of the accepted request (@authority, the signature-agent member, @method, @path and the content digest) equals the value the Agent signed, so no covered value is substituted. The attacker has the network only.	`network`	proved	strong	checked
D01.8	With the minimal coverage set, the components it does cover (@authority and the signature-agent member) still equal the value the Agent signed, so the identifier cannot be substituted. The attacker has the network only.	`network`	proved	bind	checked
U01	With the stronger coverage set any substitution of method, target URI or body makes verification fail, so the accepted request carries the values the Agent signed. The attacker has the network only.	`network`	proved	strong	checked
U01	With the minimal coverage set, a captured signature is accepted against a different method, path and body on the same authority. The attacker has the network only: it records one request and changes the uncovered fields.	`network`	attack (holds_in_source)	min (the control of strong)	
U01	Even with the stronger coverage set the same signature message instance is accepted more than once within its lifetime. The attacker has the network only: it records one request and replays it unchanged.	`network`	attack (holds_in_source)	replay	

target	claim	attacker	answer	query	control
X01	Requests from one Agent cannot be correlated across origins by the identifier they carry: no two Origins ever attribute requests to the same (Signature-Agent URL, keyid) identifier. The attacker has the network only.	`network`	attack (holds_in_source)	unlink	
X01	Under one Signature-Agent URL every signed request of an Agent carries the same keyid, which is what makes the correlation possible. The attacker has the network only.	`network`	proved	sameid	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-20261114T182048, 2026-11-14. Status: finished. Model: deepseek-v4-1-flash-261017. Steps: 51, prover 6 s, wall 12 min, cost 0.110 USD.

ProVerif: 90 proofs run, 90 decided, 0 without an answer, 3 s, 29 answers recorded.

What was analysed

Protocol	WEBBOTAUTH: signed requests, key discovery and identity binding
Version	draft-ietf-webbotauth-httpsig-protocol-01
Analysed	Verification directly by the Origin; a single Signature-Agent HTTPS URL; a fixed signing algorithm; first the request authentication without caching, proxies or redirects. The minimal compliant signature coverage set and the stronger coverage set are each kept; recommended fields cannot be turned into mandatory ones.
Not analysed	The main draft defines the full application interaction, but the signature serialisation/verification algorithm is still defined by RFC 9421; the minimal compliant configuration allows signature reuse at the @authority level; behaviour the reproduction draft already describes does not count as a new vulnerability; request authentication is not equated with user authorisation, delegation or session establishment; cache invalidation/real-time windows are abstracted separately as bounded. Only the properties under the explicitly modelled conditions and attacker are proved; missing checks, a full secure channel or an all-honest NF cannot substitute for the conclusions to be proved.
Source text	785 numbered passages (159459 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	Identifiers and Trust Model
main	4.1	The Signature-Agent URL is the identifier
main	4.2	Rotation
main	4.3	When No URL Is Available
main	4.4	What the URL Endorses
main	4.5	Binding a Key to a Web Origin
main	4.6	Out of Scope
main	5	Protocol Overview
main	5.1	Deployment Models
main	5.2	Generating HTTP Message Signature
main	5.2.1	Signature-Agent
main	5.2.2	Multiple Signatures
main	5.2.3	Replay Protection
main	5.2.4	Additional Headers

Document	Section	Title
main	5.2.5	Sending a Request
main	5.3	Requesting a Message Signature
main	5.4	Validating Message Signature
main	5.5	Key Distribution and Discovery
main	5.5.1	Directory Format
main	5.5.2	Key Rotation
main	5.5.3	Redistributed Key Material
main	5.6	Session Considerations
main	6	Security Considerations
main	6.1	Use of TLS
main	6.2	Performance Impact
main	6.3	Key Compromise Response
main	6.4	Shared Secrets Considered Harmful
main	6.5	Key Reuse Considered Harmful
main	6.6	Reverse Proxy Consideration
main	6.6.1	Signature-Agent Labeling
main	6.7	Server-Side Request Forgery (SSRF)
main	6.8	Test and Demonstration Keys
main	6.9	Static Signatures
main	6.10	Discovery Failure
main	6.11	Unsigned Requests
rfc9421	2	HTTP Message Components
rfc9421	2.1	HTTP Fields
rfc9421	2.1.1	Strict Serialization of HTTP Structured Fields
rfc9421	2.1.2	Dictionary Structured Field Members
rfc9421	2.1.3	Binary-Wrapped HTTP Fields
rfc9421	2.1.4	Trailer Fields
rfc9421	2.2	Derived Components
rfc9421	2.2.1	Method
rfc9421	2.2.2	Target URI
rfc9421	2.2.3	Authority
rfc9421	2.2.4	Scheme
rfc9421	2.2.5	Request Target
rfc9421	2.2.6	Path
rfc9421	2.2.7	Query
rfc9421	2.2.8	Query Parameters

Document	Section	Title
rfc9421	2.2.9	Status Code
rfc9421	2.3	Signature Parameters
rfc9421	2.4	Signing Request Components in a Response Message
rfc9421	2.5	Creating the Signature Base
rfc9421	3	HTTP Message Signatures
rfc9421	3.1	Creating a Signature
rfc9421	3.2	Verifying a Signature
rfc9421	3.2.1	Enforcing Application Requirements
rfc9421	3.3	Signature Algorithms
rfc9421	3.3.1	RSASSA-PSS Using SHA-512
rfc9421	3.3.2	RSASSA-PKCS1-v1_5 Using SHA-256
rfc9421	3.3.3	HMAC Using SHA-256
rfc9421	3.3.4	ECDSA Using Curve P-256 DSS and SHA-256
rfc9421	3.3.5	ECDSA Using Curve P-384 DSS and SHA-384
rfc9421	3.3.6	EdDSA Using Curve edwards25519
rfc9421	3.3.7	JSON Web Signature (JWS) Algorithms
rfc9421	7	Security Considerations
rfc9421	7.1	General Considerations
rfc9421	7.1.1	Skipping Signature Verification
rfc9421	7.1.2	Use of TLS
rfc9421	7.2	Message Processing and Selection
rfc9421	7.2.1	Insufficient Coverage
rfc9421	7.2.2	Signature Replay
rfc9421	7.2.3	Choosing Message Components
rfc9421	7.2.4	Choosing Signature Parameters and Derived Components over HTTP
rfc9421	7.2.5	Signature Labels
rfc9421	7.2.6	Multiple Signature Confusion
rfc9421	7.2.7	Collision of Application-Specific Signature Tag
rfc9421	7.2.8	Message Content
rfc9421	7.3	Cryptographic Considerations
rfc9421	7.3.1	Cryptography and Signature Collision
rfc9421	7.3.2	Key Theft
rfc9421	7.3.3	Symmetric Cryptography
rfc9421	7.3.4	Key Specification Mixup
rfc9421	7.3.5	Non-deterministic Signature Primitives
rfc9421	7.3.6	Key and Algorithm Specification Downgrades

Document	Section	Title
rfc9421	7.3.7	Signing Signature Values
rfc9421	7.4	Matching Signature Parameters to the Target Message
rfc9421	7.4.1	Modification of Required Message Parameters
rfc9421	7.4.2	Matching Values of Covered Components to Values in the Target
rfc9421	7.4.3	Message Component Source and Context
rfc9421	7.4.4	Multiple Message Component Contexts
rfc9421	7.5	HTTP Processing
rfc9421	7.5.1	Processing Invalid HTTP Field Names as Derived Component Names
rfc9421	7.5.2	Semantically Equivalent Field Values
rfc9421	7.5.3	Parsing Structured Field Values
rfc9421	7.5.4	HTTP Versions and Component Ambiguity
rfc9421	7.5.5	Canonicalization Attacks
rfc9421	7.5.6	Non-List Field Values
rfc9421	7.5.7	Padding Attacks with Multiple Field Values
rfc9421	7.5.8	Ambiguous Handling of Query Elements

The purpose of the protocol as the source step located it in the documents: verification directly by the Origin; a single Signature-Agent HTTPS URL; a fixed signature algorithm; request authentication first, with no caching, no proxying, and no redirects. Keep the minimum compliant signature coverage set and the stronger coverage set separately, and do not turn the recommended fields into mandatory requirements.

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
D01	signature attributed to resolved URL	the Origin: an accepted web-bot-auth signature was produced by a holder of the key published at the Signature-Agent URL the Origin resolved (or, in the fallback, by the holder of the keyid thumbprint), over the covered components of the request it received	Main 4.4 says a valid signature over a resolved URL proves that a holder of a key that URL publishes signed the covered message, and a verifier MUST NOT attribute a request to a URL unless it made the URL-key association; Main 5.2.2 says each signature is validated independently against its own covered components and key.	s0006, s0007, s0008, s0021, s0064, s0070, s0074, s0111
A03	identifier matches a real Agent run	the Origin: the identifier it attributes an accepted request to (the Signature-Agent URL, or the keyid thumbprint where it holds the key by other means) is one under which an Agent really signed those covered values	Main 4.1: the identifier is the URL the verifier fetched and what the verifier ends up with is a pair, that URL and a key the URL provides; an unresolved Signature-Agent value is a claim, not an identity.	s0006, s0007, s0008, s0020, s0111

ID	Name	About	What the documents say	Where
A07	URL binds the key it publishes	the Origin and the party owning a Signature-Agent URL: the URL an accepted request is attributed to is the URL that published the verifying key, and a request signed with a key published at one URL is never attributed to another	Main 4.4 and 4.5: the identifier is the URL and the association URL-keys is what makes it usable; Main 4.1 says keyid cannot carry identity across a rotation; Main 5.4 requires key lookup keyed on the (URL, key) pair so a key learned from one URL is not attributed to another.	s0006, s0011, s0111, s0122
R02	one acceptance per signature instance	the Origin: within the signature's expires window the same signature instance is accepted at most once, and a substituted request cannot reach beyond the components the signature covers	Main 5.2: a signature covering @authority alone verifies against any method, path or body until it expires, so anyone who observes one request can reuse it against the same origin; Main 5.2.3 defines no additional nonce requirements and defers replay protection to RFC 9421 7.2.2; Main 6.9 says a precomputed signature must not be treated as a long-lived credential.	s0051, s0076, s0188, s0189
K01	signing key leak boundary	the Agent's signing private key: it is kept from the attacker unless exfiltrated, and once it leaks the attacker can act as that Agent until every verifier has refreshed its directory and the signatures have expired	Main 5.2: the signing key is available to the agent at request time; Main 6.3: no revocation, removal from the directory is the only remedy and takes effect at the next refresh; RFC 9421 7.3.2: a stolen signing key lets the attacker generate their own signatures.	rfc9421:3605, s0049, s0139, s0159, s0160
R05	pinned algorithm and key use	the Origin: it accepts only the fixed asymmetric signature algorithm with a key published at the URL, never a shared-secret HMAC or a downgraded algorithm, and an Agent's signing key is not reused for another purpose	Main 6.4: implementations MUST NOT use shared HMAC because it breaks non-repudiation; Main 6.5: implementations SHOULD NOT reuse a signing key for different purposes; RFC 9421 7.3.6 and 7.3.4 describe key and algorithm specification downgrades and mixups.	rfc9421:3672, rfc9421:3646, s0162, s0164, s0151, s0047
A07.7	key lookup keyed on URL	the Origin: the URL it attributes an accepted signature to is the URL that published that very key, because it looks the key up on the (URL, key) pair and not on the keyid alone	Main 5.4: key lookup MUST be keyed on the (URL, key) pair, not on the key alone; a verifier that indexes by keyid alone will verify a request as coming from one URL using a key it learned from another and attribute it to the URL the client asserted, and the party whose URL is asserted cannot detect or stop it.	s0111, s0006, s0022, s0122
D01.8	covered components match the request	the Origin: every component the signature covers carries, in the request it verifies, exactly the value the Agent signed, namely @authority, the signature-agent member and, with the stronger set, @method, @path and the Content-Digest	Main 5.2 lists the components the Agent covers and the stronger set it may add, and says no component covers the body unless Content-Digest is sent and covered; RFC 9421 3.2 has the verifier derive the signature base from the received message, and 7.2.8 requires the Content-Digest value itself to be validated against the content.	rfc9421:3784, rfc9421:3547, s0052, s0060, s0538, s0659
U01	coverage sets and replay boundary	the Origin and the Agent: with the minimal compliant coverage set a captured signature can be replayed against a different method, path or body on the same authority until it expires; with the stronger set any such replacement fails verification, though an identical request can still be replayed	Main 5.2: a signature covering @authority alone verifies against any method, path or body until it expires; the @method, @path and @target-uri components narrow the scope but a signer that omits them remains conformant; Main 6.9: bounded created and expires reduce replay risk but do not remove it.	s0051, s0052, s0053, s0058, s0060, s0076, s0188, s0189

ID	Name	About	What the documents say	Where
X01	requests from one Agent are unlinkable	the Agent's privacy: the URL and keyid identifier by which a receiver recognises the Agent across requests	Main 7.1 states that the protocol enables receivers to recognise requests that carry the same identifier and to correlate them, including across origins, and that it is unsuitable for agents that need to avoid such identification; key rotation does not prevent the correlation while the identifier is unchanged.	main:1238

Added from the documents by the analyst: X01.

The security statements of the documents, paragraph by paragraph

The host listed the 298 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: 157. Not taken: 141. 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 to P005	main:1024-1046	R02, U01, `network`	
P006	main:1051-1058	No goal stated	performance of signature verification and key caching; it states no property of the protocol
P007 to P008	main:1062-1071	K01, `key_theft`	
P009 to P010	main:1075-1087	R05	
P011 to P014	main:1091-1109	R02, A07.7	
P015 to P017	main:1113-1126	A07.7	
P018 to P027	main:1130-1167	The prover cannot be asked	server-side request forgery: response size, key count, fetch latency, redirect depth and address ranges. These are resource and network-policy limits of the fetch, not a property of the messages a prover can be asked about
P028	main:1171-1174	R05	
P029 to P030	main:1178-1185	R02, U01	
P031 to P032	main:1189-1206	A07.7	
P033 to P036	main:1208-1225	Outside the scope	the rule that a resolver failure must not evict a cached association, and the correlation of outages, is about the verifier's cached directory state; the cache is an abstraction the task leaves to a bounded model and it is not modelled here
P037	main:1229-1232	No goal stated	it says an unsigned request establishes nothing, and that origins decide how to treat it: no property must hold
P038 to P039	main:1238-1246	X01, `network`	
P040	main:1250-1255	No goal stated	advice on what a key directory should contain and on logging directory requests; it states no property of the exchange

Paragraphs	Where	Taken up as	Why not
P041	rfc3986:2223-2228	Outside the scope	generic URI security; the model treats the Signature-Agent URL as an opaque identifier
P042	rfc3986:2232-2242	A07.7	
P043 to P056	rfc3986:2246-2392	Outside the scope	URI construction, transcoding, rare IP address forms and userinfo display attacks: parsing of the URL, not the signature or discovery protocol; the model treats the URL as opaque
P057 to P063	rfc6585:276-310	The prover cannot be asked	status codes 428, 429, 431 and 511: rate limiting, request sizes and captive portals, i.e. availability behaviour
P064	rfc7517:997-1000	No goal stated	general statement that JWK agents must address cryptographic security issues; no property of this protocol
P065 to P070	rfc7517:1008-1044	A07.7	
P071 to P072	rfc7517:1048-1058	K01, `key_theft`	
P073	rfc7517:1062-1073	No goal stated	RSA private key representation and blinding: an implementation detail of the primitive
P074	rfc7517:1077-1078	The prover cannot be asked	key entropy and random number generation: the strength of a primitive, which a trace prover cannot be asked
P075 to P078	rfc7638:521-556	A03, A07.7	
P079	rfc7638:558-566	R05	
P080 to P083	rfc8037:245-262	R05	
P084	rfc8037:272-276	The prover cannot be asked	random number generation for key generation: the strength of a primitive
P085	rfc8037:278-280	Outside the scope	X25519/X448 key exchange, which this protocol does not use
P086 to P087	rfc9110:8464-8479	No goal stated	an introduction to HTTP's security considerations; it states no property of this protocol
P088 to P094	rfc9110:8483-8533	A07.7	
P095 to P098	rfc9110:8537-8556	R02, A07.7	
P099 to P146	rfc9110:8560-8967	Outside the scope	file and path names, command and query injection, element lengths, compression, personal information, log privacy, field names, fragments, product information, fingerprinting, range denial of service and HTTP's own authentication framework: server-side application and transport concerns outside the message flow analysed here
P147 to P151	rfc9111:1572-1603	K01, `key_theft`	
P152 to P156	rfc9111:1607-1633	Outside the scope	cache timing attacks and caching of sensitive information: HTTP caching behaviour, not the directory association the protocol defines

Paragraphs	Where	Taken up as	Why not
P157 to P165	rfc9421:3282-3306	D01, R05, R02, U01	
P166 to P167	rfc9421:3312-3326	The prover cannot be asked	the risk that a deployment switches signature verification off: a deployment practice, not a trace property of the protocol
P168 to P170	rfc9421:3330-3342	R02, `network`	
P171 to P172	rfc9421:3348-3364	D01.8, U01	
P173 to P175	rfc9421:3368-3391	R02, U01	
P176 to P179	rfc9421:3395-3418	U01, D01.8	
P180 to P183	rfc9421:3421-3445	D01.8, U01	
P184 to P185	rfc9421:3449-3461	No goal stated	it says applications should put no meaning on signature labels; nothing must hold, and the model does not distinguish labels
P186 to P188	rfc9421:3465-3485	D01, A07	
P189 to P190	rfc9421:3489-3507	R05, D01.8	
P191 to P204	rfc9421:3511-3568	D01.8, U01	
P205 to P208	rfc9421:3574-3601	R05	
P209 to P210	rfc9421:3605-3617	K01, `key_theft`	
P211 to P213	rfc9421:3621-3642	R05	
P214	rfc9421:3646-3654	R05, D01	
P215	rfc9421:3658-3668	The prover cannot be asked	non-deterministic signature primitives: the model uses a deterministic signature function and the paragraph is about how a verifier must call the primitive
P216 to P217	rfc9421:3672-3691	R05	
P218 to P231	rfc9421:3695-3758	D01.8	
P232 to P233	rfc9421:3764-3779	The prover cannot be asked	denial of service through modified required parameters, and the risk of relaxing requirements: availability and deployment behaviour
P234 to P240	rfc9421:3782-3839	D01.8	
P241 to P243	rfc9421:3843-3875	A07.7, D01.8	

Paragraphs	Where	Taken up as	Why not
P244 to P255	rfc9421:3881-4002	Outside the scope	field-name injection, semantically equivalent field values, structured-field parsing, HTTP version ambiguity and canonicalization: the signature-base serialization owned by RFC 9421; the model derives the base from typed values
P256 to P268	rfc9421:4006-4086	Outside the scope	non-list field values, Set-Cookie combining and padding attacks: byte-level field processing in the signature base, outside the model of the message flow
P269 to P270	rfc9421:4090-4108	U01	
P271 to P272	rfc9421:4114-4129	X01, `network`	
P273 to P274	rfc9421:4133-4140	No goal stated	signatures provide no confidentiality: the protocol claims none, and no target asks for it
P275	rfc9421:4144-4152	No goal stated	error messages acting as oracles: advice on what a server should not reveal, with no property of the exchange modelled
P276 to P277	rfc9421:4156-4172	D01.8	
P278 to P286	rfc9530:564-628	D01.8, U01	
P287 to P289	rfc9530:632-648	Outside the scope	integrity fields in trailer sections, which this protocol does not use
P290 to P291	rfc9530:652-664	D01.8	
P292 to P294	rfc9530:668-689	R05	
P295	rfc9530:693-698	The prover cannot be asked	resource exhaustion from validating integrity fields: availability
P296	rfc9651:1741-1745	The prover cannot be asked	size limits of structured fields: resource consumption
P297 to P298	rfc9651:1747-1757	Outside the scope	injection of new HTTP fields changing the meaning of a structured field, and Display String sanitisation: parsing of the header serialization, owned by RFC 9651 and not modelled

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: 1. Without: 22. 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	Agent	party	Nothing gained	a controlled Agent holds only its own signing key and can only be attributed to the URL it controls; that is the identifier working as intended
S02	Origin	party	Nothing gained	the Origin holds only public key material; a controlled verifier is the party in whose favour the targets are stated, and the documents ask nothing of a dishonest one

ID	Party or value	Kind	Attacker	Why none
S03	Directory	party	Nothing gained	the Directory is modelled as honest and is reached over the TLS-authenticated fetch; an operator that controls a directory speaks only for its own URL, which is exactly the association the Origin checks, and it still cannot sign for a key it does not hold
S04	Attacker	party	Nothing gained	the party is the adversary itself, already present as the network attacker; controlling it adds nothing beyond the network
S05	Agent signing private key	value	`key_theft`	
S06	Binding between the Signature-Agent URL and the key	value	Nothing gained	these are public values: the URL, the keyid thumbprint, the published key set, the signature and the covered components all travel on the wire and the attacker on the network already holds them; the only secret in the protocol is the Agent's signing key, listed separately
S07	Covered request components	value	Nothing gained	these are public values: the URL, the keyid thumbprint, the published key set, the signature and the covered components all travel on the wire and the attacker on the network already holds them; the only secret in the protocol is the Agent's signing key, listed separately
S08	Signature-Agent URL identifier	value	Nothing gained	these are public values: the URL, the keyid thumbprint, the published key set, the signature and the covered components all travel on the wire and the attacker on the network already holds them; the only secret in the protocol is the Agent's signing key, listed separately
S09	keyid thumbprint identifier	value	Nothing gained	these are public values: the URL, the keyid thumbprint, the published key set, the signature and the covered components all travel on the wire and the attacker on the network already holds them; the only secret in the protocol is the Agent's signing key, listed separately
S10	Covered component values	value	Nothing gained	these are public values: the URL, the keyid thumbprint, the published key set, the signature and the covered components all travel on the wire and the attacker on the network already holds them; the only secret in the protocol is the Agent's signing key, listed separately
S11	Binding between the URL and the key	value	Nothing gained	these are public values: the URL, the keyid thumbprint, the published key set, the signature and the covered components all travel on the wire and the attacker on the network already holds them; the only secret in the protocol is the Agent's signing key, listed separately
S12	keyid thumbprint	value	Nothing gained	these are public values: the URL, the keyid thumbprint, the published key set, the signature and the covered components all travel on the wire and the attacker on the network already holds them; the only secret in the protocol is the Agent's signing key, listed separately
S13	JWKS directory contents	value	Nothing gained	these are public values: the URL, the keyid thumbprint, the published key set, the signature and the covered components all travel on the wire and the attacker on the network already holds them; the only secret in the protocol is the Agent's signing key, listed separately
S14	Signed message instance	value	Nothing gained	these are public values: the URL, the keyid thumbprint, the published key set, the signature and the covered components all travel on the wire and the attacker on the network already holds them; the only secret in the protocol is the Agent's signing key, listed separately
S15	created/expires time window	value	Nothing gained	these are public values: the URL, the keyid thumbprint, the published key set, the signature and the covered components all travel on the wire and the attacker on the network already holds them; the only secret in the protocol is the Agent's signing key, listed separately

ID	Party or value	Kind	Attacker	Why none
S16	Set of covered components	value	Nothing gained	these are public values: the URL, the keyid thumbprint, the published key set, the signature and the covered components all travel on the wire and the attacker on the network already holds them; the only secret in the protocol is the Agent's signing key, listed separately
S17	Signature algorithm choice	value	Nothing gained	these are public values: the URL, the keyid thumbprint, the published key set, the signature and the covered components all travel on the wire and the attacker on the network already holds them; the only secret in the protocol is the Agent's signing key, listed separately
S18	Signature key usage	value	Nothing gained	these are public values: the URL, the keyid thumbprint, the published key set, the signature and the covered components all travel on the wire and the attacker on the network already holds them; the only secret in the protocol is the Agent's signing key, listed separately
S19	Signature-Agent URL and directory	value	Nothing gained	these are public values: the URL, the keyid thumbprint, the published key set, the signature and the covered components all travel on the wire and the attacker on the network already holds them; the only secret in the protocol is the Agent's signing key, listed separately
S20	URL-to-key binding	value	Nothing gained	these are public values: the URL, the keyid thumbprint, the published key set, the signature and the covered components all travel on the wire and the attacker on the network already holds them; the only secret in the protocol is the Agent's signing key, listed separately
S21	Signature	value	Nothing gained	these are public values: the URL, the keyid thumbprint, the published key set, the signature and the covered components all travel on the wire and the attacker on the network already holds them; the only secret in the protocol is the Agent's signing key, listed separately
S22	Set of components covered by the signature	value	Nothing gained	these are public values: the URL, the keyid thumbprint, the published key set, the signature and the covered components all travel on the wire and the attacker on the network already holds them; the only secret in the protocol is the Agent's signing key, listed separately
S23	Request method, target-uri, body	value	Nothing gained	these are public values: the URL, the keyid thumbprint, the published key set, the signature and the covered components all travel on the wire and the attacker on the network already holds them; the only secret in the protocol is the Agent's signing key, listed separately

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		
`key_theft`	the Agent's signing private key from the moment it is exfiltrated; the model gives it to the attacker outright and the verifier keeps accepting signatures made with it for as long as its cached directory is not refreshed and the signature has not expired	RFC 9421 7.3.2: if the keys used to sign are stolen the attacker can generate their own signatures. Main 6.3: the document defines no revocation; removing the key from the directory is the only remedy and takes effect at each verifier on its next refresh, so the key keeps verifying until then.	rfc9421:3605, main:1062

ID	network	key_theft
D01	Holds	
A03	Holds	
A07	Holds	
R02	Fails	
K01	Holds	Fails
R05	Holds	
A07.7	Holds	
D01.8	Holds	
U01	Fails	
X01	Fails	

Source and targets

Source: 785 units, 159459 bytes, frozen in source-delivery-01.json. Targets: 7 approved (D01, A03, A07, R02, K01, R05, U01); the questions asked: 3. The list of targets was approved as drafted, beforehand, with --approve-targets.

Decisions the host took by its documented default (task `decisions: auto`), for you to check:

- approve_package: approved source package protocol-34ea382432f4dc0820469112d1c6

This step as it reports itself: 2 requests (2 failed), 0.000 USD, 165429 tokens in and 9920 out, 56 s.

Notes on the prover and the model

What the analyst named for targets:

step	targets	features	reason
35	D01, A03, A07, A07.7, D01.8, R02, R05, U01, K01, X01	plain_primitives, many_sessions	The model uses only a deterministic signature scheme (sign/verify over a typed signature base), a hash for the content digest and a keyid thumbprint function, with no algebraic laws; the TLS-authenticated directory fetch is modelled as a private channel between the Origin and the host named in the URL. Every role is replicated so the answers cover any number of sessions, agents and two origins. How long a removed key or a captured signature keeps verifying (directory cache lifetime, expires) is not modelled at all.

The analyst's closing note

Ten targets, 23 recorded answers, all re-checked on the final model; no falsified query left unsubmitted and no target without an answer under the attacker it names. Three documented weaknesses are reported as `recorded` results with figures; there is no new finding.

Established (network attacker only, unless stated): - Attribution (D01, A03, A07, A07.7, R05): every request the Origin accepts was signed by a holder of a key published at the Signature-Agent URL it resolved, and the attributed URL is the URL that published that very key, because the lookup is on the (URL, key) pair. Controls: the same query without the no-leak exception, and a lax verifier that takes the key set from anyone, are both falsified. - Component binding (D01.8, U01): with the strong coverage set every covered component equals what the Agent signed; with the minimal set only @authority and the signature-agent member are bound. - Replay (R02, U01): one recorded signature is accepted twice by two Origin sessions; the strong coverage set does not help. Documented (5.2.3, 6.9;

RFC 9421 7.2.2). - Minimal coverage substitution (U01): a signature over @authority alone is accepted for another method, path and body. Documented (5.2). - Key theft (K01, attacker with the signing key): without a leak the key stays secret; with it the adversary is attributed to the Agent's URL with no Agent run, until verifiers refresh their directory. Documented (6.3; RFC 9421 7.3.2). - Algorithm pin (R05): no acceptance with a lowered or shared-secret algorithm; the lax-verifier control shows the HMAC-with-public-key downgrade goes through where the algorithm comes from the message. - Correlation (X01): two origins attribute requests to the same URL and keyid, so an Agent is correlatable across sites. Documented (7.1).

Remedies checked: covering @method, @path and Content-Digest removes the substitution (attack query verified, all other answers unchanged). A nonce plus a table of accepted nonces did NOT remove the replay: two sessions pass the check before either records the nonce, so an implementation needs an atomic test-and-record, which this model cannot express. The revocation remedy for key theft cannot be modelled because the draft defines no revocation mechanism.

What blocks more: the directory cache, key rotation/removal and the created/expires window are not modelled, so no duration is established for the replay or the compromise window; redistribution and the Appendix B directory response signature are out of scope; the TLS-authenticated fetch is an assumption and the signature is ideal. A bounded model of the cache and rotation, or Tamarin+ Agent, would settle the durations.