Security analysis of ADRF subscribe and store analytics

This report in another language: 中文.

19.7.0 · ProVerif+ Agent report

Documents	5G System; Analytics Data Repository Services; Stage 3 (protocol definition); 5G System; Network Data Analytics Services; Stage 3 (supporting)
Properties examined	11: 10 hold, 1 fail (1 attack)
Findings	2

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 ADRF data management service lets an NF service consumer have data or analytics collected and stored on its behalf: the consumer sends a StorageSubscriptionRequest, the ADRF subscribes to the NWDAF named as targetNfId, stores the notifications the NWDAF sends to the notification URI it gave out, and removes the upstream subscription when the consumer sends a StorageSubscriptionRemoval. The parties are the NF service consumer, the ADRF, the upstream NWDAF and the NRF as authorization server. The consumer holds its NF credentials and an OAuth2 token with the storage scope, the ADRF holds its own credentials and the mapping between the transRefIds it assigns and the upstream subscriptions, and the NWDAF holds the subscription identifier and the notification URI it was given.

When the exchange ends the ADRF is sure of the peer for every subscribed response and notification and of the parameters of the subscription it recorded, the consumer is sure that its own request created the transaction and holds the transRefId, and the NWDAF is sure that the ADRF, under a token for the event subscription scope, asked for the subscription it holds. Every message travels under TLS between the endpoint identities, and each operation is accepted only with a token whose subject is the endpoint that sent it.

The parties:

- **NF service consumer**: requester of collection and storage. Holds: NF credentials; OAuth2 token: subject consumer, audience ADRF, scope nadrf-datamanagement:storage-read-delete-subs.
- **ADRF**: service producer, stores the analytics. Holds: its NF credentials; the transRefIds it assigns and their upstream subscriptions.
- **upstream NWDAF**: analytics producer. Holds: its NF credentials; the subscriptionIds it creates.
- **NRF**: authorization server. Holds: the NF credentials and the authorization server role.

Figure 1 shows the exchange as the documents specify it.

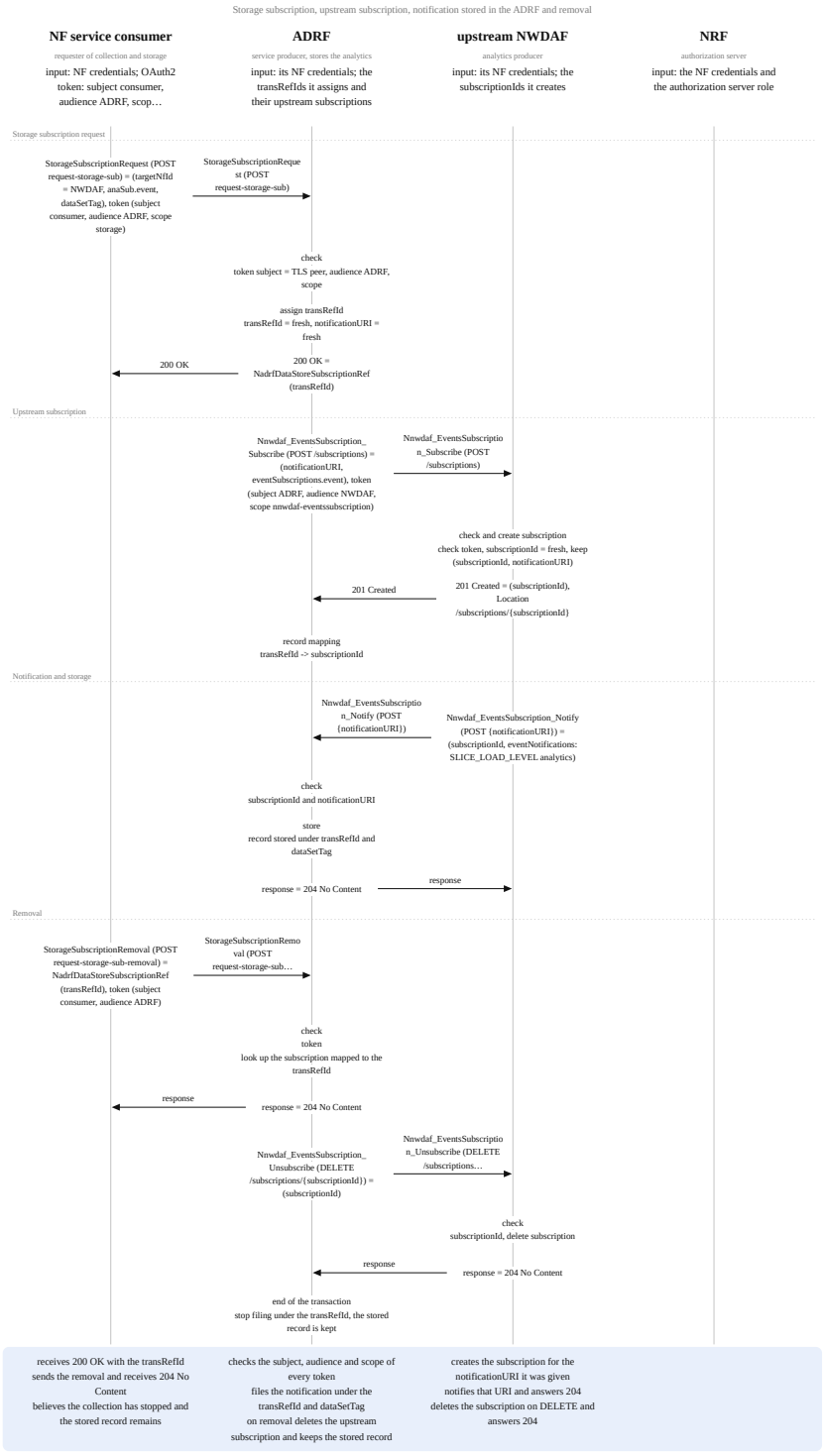


Figure 1. The protocol as specified.

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

The analysis has 2 findings: what follows is not stated in the documents. For 2 of them it is unclear whether the documents state them.

The API sub-flow does what it is meant to do against an attacker on the network, with one weakness: the ADRF removes an upstream subscription on any valid token of the API once a removal request names the transRefId, so a consumer that knows another consumer's transRefId can stop that consumer's collection. Everything else asked holds in the model, including the peer authentication of the upstream NWDAF, the agreement on the subscription parameters, the binding of the stored record to the consumer's request, the subject, audience and scope binding of the tokens, and the lifecycle after a removal. The second weakness is read from the source rather than shown by a query: the storage subscription is confirmed before the upstream subscription exists, with no rollback. The answers rest on a model where TLS is an idealised key per endpoint pair, OAuth2 has no expiry, the ADRF lifecycle is unrolled to three steps, and the sharing of one upstream subscription by several transRefIds is not modelled.

The attack figures use the columns of Figure 1. Red is what the attacker does. Yellow marks a message that a party sends although the specified exchange does not have it at that point, and the outcome that the attack breaks.

2.1 A known transRefId lets any authorized NF stop another consumer's collection

Kind	Weakness of the protocol as specified
New	Unclear: the documents state it in part, or could not be searched
Value	medium
Properties	Z03 (section 3)
Source text	s0034, s0035, s0293
Shown by	ProVerif attack query `z03g`; Figure 2

What happens. A removal request names a transRefId and the ADRF removes the upstream subscription mapped to it, as s0035 requires, but nothing compares the consumer recorded for that transRefId with the subject of the request token. A second NF service consumer that knows the identifier therefore has the first consumer's upstream subscription deleted and its collection stopped.

What the attacker needs. An authorized NF service consumer acting under its own valid credentials, and the transRefId of another consumer's storage subscription. That identifier travels only between the ADRF and its owner, so the attacker needs it from a log, from the owner, or from the error answer to a guess.

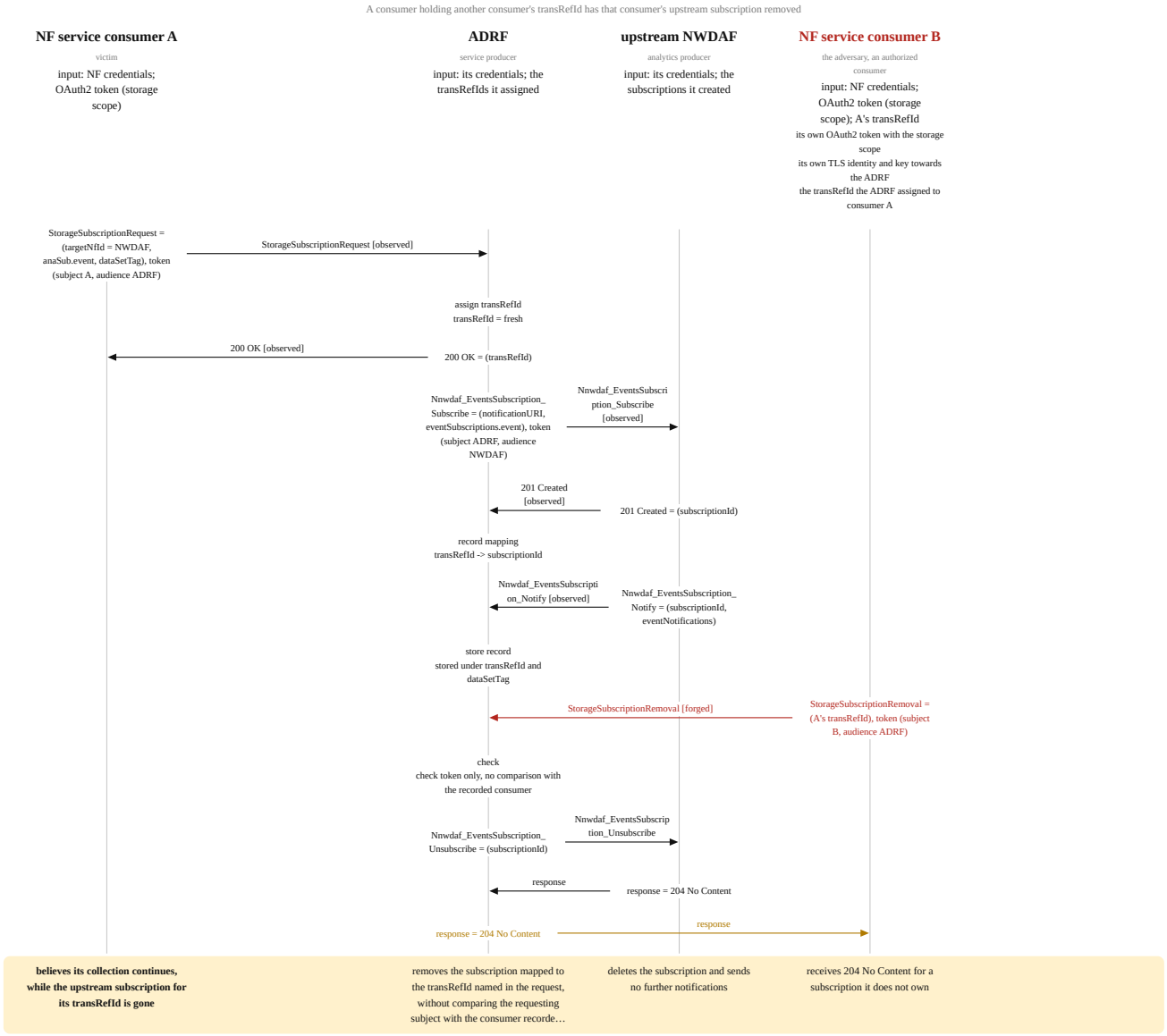


Figure 2. A consumer holding another consumer's transRefId has that consumer's upstream subscription removed. Property Z03; attack query z03g.

Proposed change. The ADRF shall record the NF instance identity of the consumer that created each transRefId, and on a StorageSubscriptionRemoval request verify that the token subject and the TLS peer identity are that consumer before removing the mapped upstream subscription; a removal naming another consumer's transRefId is answered 403 Forbidden. Checked on a copy of the unrolled model: the attack query is verified there and the other recorded answers still hold.

Check of the change with ProVerif. The change was put into a copy of the model: The ADRF records the NF instance identity of the consumer that created each transRefId, and on a StorageSubscriptionRemoval request it verifies that the token subject (and TLS peer identity) is that recorded consumer; a removal naming a transRefId of another consumer is answered with 403 Forbidden and nothing is removed. On the changed model ProVerif proves `z03g`: the attack is no longer possible there. The other 3 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-owner.pv`.

2.2 A storage subscription is confirmed before its upstream subscription exists, with no rollback

Kind	Weakness of the protocol as specified
New	Unclear: the documents state it in part, or could not be searched
Value	medium
Properties	A08.8 (section 3)
Source text	s0021, s0023, s0026

What happens. The ADRF assigns the transRefId and answers 200 OK before it creates the upstream subscription, and the source gives no rule for a creation that fails afterwards. The consumer then holds a transRefId whose subscription never existed, and the ADRF holds a mapping to nothing.

What the attacker needs. Nothing beyond the network and an upstream creation the NWDAF rejects or that is lost, both of which the source allows (400 Bad Request, 500 Internal Server Error).

Proposed change. The ADRF should answer 200 OK with the transRefId only after the upstream subscription exists, or report the failed creation to the consumer in a notification correlated by notifCorrId.

Check of the change with ProVerif. Not checked: The property is that a failed upstream creation is rolled back and reported; no recorded query shows it, because the source states no rollback rule at all and a query could only restate a choice the model makes.

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	**upstream peer authentication at ADRF.** Whenever the ADRF accepts a subscription response or event notification from an upstream NWDAF, there exists a really running upstream NWDAF whose intended peer identity is that ADRF.	no compromise	Holds	medium	no
A03	**agreement on upstream subscription parameters.** Whenever the ADRF believes that an upstream subscription has been established with a given set of parameters, there exists a really running upstream NWDAF whose role, identity and subscription parameters (anaSub/dataSub, targetNfId, formatInstruct/procInstruct, storeHandl, dataSetTag) agree with those recorded by the ADRF.	no compromise	Holds	low	no
A08	**stored record bound to requester request.** The notifCorrId, dataSetTag/dataSetId and targetNfId carried by a notification that the ADRF accepts and stores must correspond to the values the requester gave in that StorageSubscriptionRequest, and must be consistent with the current security context (TLS endpoint identity, subject/audience/scope of the OAuth2 token).	no compromise	Holds	medium	no
K01	**OAuth2 token secrecy.** An attacker cannot derive the OAuth2 access tokens or NF credentials held by the requester, the ADRF or the upstream NWDAF.	no compromise	Holds	low	no

ID	Property	Compromise assumed	Result	Value	In the documents
Z01	**authorization of API operations.** Every time the ADRF submits a storage subscription creation, a storage subscription removal or an upstream subscription creation operation, there must exist an unexpired token issued by the NRF whose subject is that requester, whose audience is the ADRF and whose scope covers that operation.	no compromise	Holds	medium	no
Z03	**removal bound to owning subject.** Every time the ADRF accepts a StorageSubscriptionRemoval and deletes the upstream subscription for a given transRefId or dataSetId, that transRefId/dataSetId must belong to the subject of the token that submitted the request.	no compromise	Fails (1 attack)	medium	unclear
S05	**lifecycle after removal.** After the ADRF has successfully processed a StorageSubscriptionRemoval, the deleted transRefId can no longer be used to create a new upstream subscription or to accept new notification storage; an upstream subscription still referenced by other active transRefIds is not deleted.	no compromise	Holds within a bound (the changing value unrolled to 3 steps)	medium	no
A08.8	**record identity correspondence.** The transRefId, dataSetTag/dataSetId and notifCorrId carried by a record the ADRF stores must correspond to the values the requester gave in that StorageSubscriptionRequest, and must correspond to the upstream subscription the ADRF created for that request.	no compromise	Holds	low	no
U01	**no cross-consumer parameter change.** For any two distinct requesters A and B, the upstream subscription parameters and storage handling parameters that the ADRF establishes or modifies for A can only be caused by A's genuine requests.	no compromise	Holds	medium	no
S05.10	**removal does not delete stored records.** After a successful StorageSubscriptionRemoval, the deleted transRefId can no longer create upstream subscriptions or accept new notification storage; stored records are not deleted by that operation, and only remove-stored-data-analytics deletes stored records.	no compromise	Holds within a bound (the changing value unrolled to 3 steps)	medium	no
X01	**confidentiality and integrity of API messages in transit.** The analytics carried in a notification from the upstream NWDAF to the ADRF, and the parameters of the storage subscription and removal requests, cannot be read or altered by an attacker on the network: no event that stores a notification uses analytics the attacker can obtain, and no operation is accepted with parameters the attacker changed.	no compromise	Holds	medium	no

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: binding of the requester, the upstream subscription and the generated ADRF record	The ADRF files a notification only under the transRefId whose upstream subscription the NWDAF reported, and under the data set tag the consumer gave in that request, so consumer, transRefId and record are tied by the ADRF's own mapping. What is missing is a rollback when the upstream subscription is never created, and the removal side is not bound to the owner, which is the finding on the transRefId.	A08.8, A08, Z03
Q2: a malicious consumer cannot change another consumer's collection/storage parameters	No: a consumer acts only under its own token and TLS identity, so it cannot create a subscription with another consumer's target, event or data set tag, nor alter the parameters recorded for another consumer. It can remove another consumer's upstream subscription if it knows the transRefId, which is the reported finding.	U01, Z01, Z03

Question	Answer	Rests on
Q3: removal of a storage subscription and the lifecycle of existing stored records must not be conflated	Yes for the modelled steps: the removal stops the upstream subscription, no stored record is deleted on that path, and a notification arriving after the accepted removal is refused. The lifecycle is answered for one transRefId per upstream subscription and three unrolled steps, so the sharing rule for further active transRefIds and any later reuse of a removed identifier stay open.	S05, S05.10

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:

- TLS is modelled as an authenticated encryption under a per-pair key derived from the two NF identities; the handshake, the certificates and their revocation are not analysed.
- OAuth2 is modelled as a token carrying subject, audience and scope; expiry, revocation and the client credential exchange with the NRF are not modelled, and the token is reusable in any number of runs.
- The ADRF's subscription state is unrolled to three steps, so the lifecycle answers hold for those steps; the rule of s0035 that an upstream subscription still mapped to further active transRefIds is not deleted is not modelled.
- Stored records, their retrieval and their deletion by remove-stored-data-analytics are outside the scope and not modelled.
- The DCCF upstream, the MFAF and the UDM consent branch are excluded, as the scope states.
- Two consumers, one ADRF and one NWDAF instance are modelled; NF set identifiers, redirects and the service communication proxy are not.
- The compromised consumer and the compromised producer are not modelled; the attacker holds only the network and a leaked transRefId.
- The strength of the MAC and encryption primitives, and resistance to quantum attack, are outside what ProVerif can be asked.

What the host records:

- The answers changed after the reading was written for: Z03.
- The answers of Z03, S05, S05.10 rest on an operation the model simplifies (Appendix D).
- Proved without a control query (Appendix C): K01: k01, 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: upstream peer authentication at ADRF

Whenever the ADRF accepts a subscription response or event notification from an upstream NWDAF, there exists a really running upstream NWDAF whose intended peer identity is that ADRF.

Every notification and every subscribed response the ADRF accepts comes from the NWDAF it subscribed to, with the subscription identifier, the event and the notification URI that NWDAF itself accepted. The guarantee rests on endpoint TLS and on the token audience, not on anything inside the payload.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value medium: It confirms mechanically the peer authentication the API text leaves to TLS and OAuth2; the proof simplifies TLS, so it is a proof of the model rather than of the deployed interface.
- Stated in the documents: no. Source text it rests on: s0026, s0913, s0961
- Rests on: TLS is modelled as authenticated encryption under a key derived from the two NF identities.; Tokens carry subject, audience and scope but no expiry.
- Open: Agreement is not injective: a repeated message of the same NWDAF run is not told from a new one.; NF set identifiers and redirects are not modelled.
- Next step: Model a consumer that puts its own notificationURI inside anaSub, to settle who chooses the callback URI.

A03: agreement on upstream subscription parameters

Whenever the ADRF believes that an upstream subscription has been established with a given set of parameters, there exists a really running upstream NWDAF whose role, identity and subscription parameters (anaSub/dataSub, targetNfId, formatInstruct/procInstruct, storeHandl, dataSetTag) agree with those recorded by the ADRF.

The parameters the ADRF records for the upstream subscription, the subscribed event and the notification URI, are the ones it sent to the NWDAF, so the subscription serving the transaction is the one the ADRF asked for.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value low: It holds because the response is authenticated; the interesting question, who chooses the callback URI, is left to the source.
- Stated in the documents: no. Source text it rests on: s0018, s0026, s0855
- Rests on: The ADRF is modelled as setting its own notification URI.
- Open: The source does not say whether the ADRF keeps its own notification URI when the consumer supplies one in anaSub.
- Next step: Settle from the source whether the ADRF overrides a notificationURI the consumer puts in anaSub, then model that case.

A08: stored record bound to requester request

The notifCorrId, dataSetTag/dataSetId and targetNfId carried by a notification that the ADRF accepts and stores must correspond to the values the requester gave in that StorageSubscriptionRequest, and must be consistent with the current security context (TLS endpoint identity, subject/audience/scope of the OAuth2 token).

A stored notification carries the data set tag of the request that created its transRefId and is filed under a subscription identifier the ADRF mapped to that transRefId, having arrived on the notification URI the ADRF gave out. The network cannot move a notification into another record.

- Result: Holds; answers what is asked: yes; usable
- Value medium: It fixes the record identity of the stored data, which the source states only as a duty to map, and it is what an implementer of the store is most likely to get wrong.
- Stated in the documents: no. Source text it rests on: s0020, s0021, s0026
- Rests on: The data set tag is an opaque value the consumer chooses.; One notification per subscription is modelled in model.pv; the unrolled model sends two.
- Open: Only the subscribed event and the data set tag represent the subscription parameters.
- Next step: Model Formatting/Processing instructions and StorageHandlingInfo to check they are bound to the same record.

K01: OAuth2 token secrecy

An attacker cannot derive the OAuth2 access token or NF credentials held by the requester, the ADRF, or an upstream NWDAF.

An attacker on the network cannot compute the token a consumer uses with the ADRF, because it is a MAC under a key derived from the consumer identity, and the interface carries it only inside the endpoint TLS sessions.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value low: The answer follows from the idealised token construction; its worth is that no message leaks the credential.
- Stated in the documents: no. Source text it rests on: s0334, s0335, s0339
- Rests on: The token is a MAC under a private per-identity key; the real token format and the NRF's role are abstracted to that.
- Open: Expiry and revocation are not modelled.; No control variant is recorded for this secrecy query, because no process gives the attacker a token.
- Next step: Model a consumer whose token is revealed, to separate token secrecy from the authority it carries.

Z01: authorization of API operations

Every time the ADRF submits a storage subscription creation, storage subscription removal, or upstream subscription creation operation, there must be an unexpired token issued by the NRF whose subject is that requester, whose audience is the ADRF, and whose scope covers the operation.

The ADRF accepts a storage subscription creation or removal only from the NF that sent it, under a token whose subject is that NF and audience the ADRF; the NWDAF accepts an upstream subscription only from the ADRF. A different NF cannot present a token for another subject, because it cannot mint one.

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

- Value medium: It shows the subject, audience and scope binding carrying the authorization of both APIs; no specification weakness was found on this path for the network attacker.
- Stated in the documents: no. Source text it rests on: s0334, s0339, s1143
- Rests on: The token is a MAC under a key derived from the subject identity, and the verifier recomputes it from the subject it also compares with the TLS peer.; Scope is one constant per API.
- Open: The compromised-consumer reading is not modelled.; Expiry and per-operation scopes are not modelled beyond one storage scope.
- Next step: Model a consumer compromised with its own credentials performing the operations of another consumer.

Z03: removal bound to owning subject

Every time the ADRF accepts a StorageSubscriptionRemoval and deletes the upstream subscription for a transRefId or dataSetId, that transRefId/dataSetId must belong to the subject of the token that submitted the request.

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

Against the network alone the ADRF removes an upstream subscription only for a removal request carrying a valid token of the consumer that created that transRefId, because the identifier cannot be guessed. Against a consumer that knows another consumer's transRefId the binding fails: the ADRF acts on any valid token, so the identifier alone decides.

- Result: Fails (1 attack); answers what is asked: yes; usable with caveats
- Value medium: One reading is proved and the other attacked with the exact steps; the value is in showing that the protection is the unguessability of an identifier the source never requires to be secret.
- Stated in the documents: unclear. Source text it rests on: s0034, s0035, s0293
- Rests on: The transRefId is modelled as an unguessable fresh name, which is what makes the network case hold.; The unrolled model carries one transRefId per upstream subscription.
- Open: The source never says whether a transRefId is a secret or a bearer capability, so the precondition of the attack is a reading of s0021 and s0293.
- Next step: Model two transRefIds mapped to one upstream subscription, to see what a removal of a shared subscription does.

S05: lifecycle after removal

After the ADRF has successfully processed a StorageSubscriptionRemoval, the deleted transRefId cannot be used to create a new upstream subscription or accept new notification storage; an upstream subscription still referenced by other active transRefIds is not deleted.

After the ADRF has accepted the removal, a notification for that subscription is not stored and a repeated removal is not acted on a second time. The state change is what makes this hold: the control variant that does not replace the kept value stores the late notification.

- Result: Holds within a bound (the changing value unrolled to 3 steps); answers what is asked: yes; usable with caveats
- Value medium: A lifecycle guarantee shown with its control, in a model that simplifies deletion and sharing; it is what a reader needs on a subscription whose removal must stop collection.
- Stated in the documents: no. Source text it rests on: s0035, s0961

- Rests on: The ADRF is unrolled to creation, first notification, removal and two later messages.; The upstream deletion is modelled as always succeeding.
- Open: The clause that an upstream subscription mapped to further active transRefIds is not deleted is not modelled.; The lifecycle is answered for three unrolled steps only.
- Next step: Unroll two transRefIds mapped to one upstream subscription to answer the sharing clause of s0035.

A08.8: record identity correspondence

The transRefId, dataSetTag/dataSetId and notifCorrId carried by the record the ADRF stores must correspond to the values the requester gave in that StorageSubscriptionRequest, and to the upstream subscription the ADRF created for that request.

The record the ADRF stores is filed under the transRefId and the upstream subscription identifier that the ADRF itself mapped to it, so the consumer's identifier, the ADRF's mapping and the record agree through the ADRF's own state.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value low: The answer is structural in this model: it shows the mapping is used consistently, not that an implementation checks it.
- Stated in the documents: no. Source text it rests on: s0021, s0026, s0035
- Rests on: Each ADRF session holds its own transRefId and subscription identifier.; Data set identifiers and their uniqueness across consumers are not modelled.
- Open: The rollback of a failed upstream creation is not modelled; it is reported as a finding instead.
- Next step: Model two consumers with the same data set tag to see how their records are told apart.

U01: no cross-consumer parameter change

For any two distinct requesters A and B, the upstream subscription parameters and storage handling parameters that the ADRF establishes or modifies for A can only be caused by A's own genuine request.

The ADRF creates a storage subscription only on a request that consumer itself sent, under its own token and TLS identity. No other consumer causes the creation, so the target, the event and the data set tag of a subscription are its own.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value medium: It separates the two consumers and shows one cannot move the other's parameters; the malicious-consumer reading is left open.
- Stated in the documents: no. Source text it rests on: s0008, s0020, s0026
- Rests on: Each consumer has its own token and TLS key.; The parameters are represented by the target, the event and the data set tag only.
- Open: The compromised-consumer reading is not modelled.; A second request for the same data set, which s0022 may serve from the existing subscription, is not modelled.
- Next step: Model a consumer compromised with its own credentials and ask whether any event of the other consumer's subscription can be produced by it.

S05.10: removal does not delete stored records

After a successful StorageSubscriptionRemoval, the deleted transRefId can no longer create an upstream subscription or accept new notification storage; stored records are not deleted by that operation, and only remove-stored-data-analytics deletes stored records.

The storage subscription removal removes the upstream subscription and nothing else: no stored record is deleted on that path and the record that was stored stays. Deleting stored data is a separate operation with its own scope.

- Result: Holds within a bound (the changing value unrolled to 3 steps); answers what is asked: yes; usable with caveats
- Value medium: It answers the confusion between stopping collection and deleting data, which an implementer of the two operations has to get right.
- Stated in the documents: no. Source text it rests on: s0035, s0186, s0215
- Rests on: The record is represented by an event the removal path never reverses.; The unrolled model covers three steps.
- Open: The remove-stored-data-analytics operation is not modelled, so the statement is shown from the removal side only.; Stored records are not modelled as retrievable objects.
- Next step: Model remove-stored-data-analytics to show the two operations act on different objects.

X01: confidentiality and integrity of API messages in transit

The analytics carried in a notification from the upstream NWDAF to the ADRF, and the parameters of the storage subscription and removal requests, cannot be read or altered by an attacker on the network: no event that stores a notification uses analytics the attacker can obtain, and no operation is accepted with parameters the attacker changed.

The analytics in a notification and the parameters of the requests are never readable or alterable by an attacker on the network: every message travels only under the keys of the two endpoint identities, and a party that legitimately holds an endpoint sees them.

- Result: Holds; answers what is asked: yes; usable
- Value medium: It is the confidentiality and integrity statement of the service based interface for this sub-flow; it holds by the TLS abstraction rather than by anything the API adds.
- Stated in the documents: no. Source text it rests on: s0913, s0961, s0334
- Rests on: TLS is a private key per ordered pair of NF identities, without a handshake.; The analytics are a fresh nonce per notification.
- Open: Compromised endpoints are not modelled, so only the network attacker is answered.; No control variant is recorded for this secrecy query.
- Next step: Model a compromised ADRF or NWDAF to see what the other party still keeps.

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		a02n	checked		2 proved
A03	proved		a03	checked		1 proved
A08	proved		a08	checked		2 proved
K01	proved		k01			1 proved
Z01	proved		z01	checked		2 proved
Z03	proved		z03	checked	z03g (holds_in_source)	1 proved, 1 attack
S05	bounded	the changing value unrolled to 3 steps	s05	checked		2 bounded
A08.8	proved		a08b	checked		1 proved
U01	proved		u01	checked		1 proved
S05.10	bounded	the changing value unrolled to 3 steps	s0510	checked		1 bounded
X01	proved		x01			1 proved

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

target	claim	attacker	answer	query	control
A02	Whenever the ADRF stores a notification for an upstream subscription, a real upstream NWDAF sent that notification for that subscription identifier: the peer that wrote the message is the NWDAF the ADRF subscribed to.	`network`	proved	a02n	checked
A02	Whenever the ADRF records an upstream subscription as established, a real NWDAF accepted that subscription with the same subscribed event and notification URI, with the ADRF as its consumer.	`network`	proved	a02s	checked
A03	The parameters under which the ADRF records the upstream subscription - the subscribed event and the notification URI - are the parameters the ADRF sent to the NWDAF.	`network`	proved	a03	checked
A08	The data set tag under which the ADRF stores a notification is the data set tag the requester gave in the storage subscription request that created that transRefId.	`network`	proved	a08	checked
A08	A notification is stored only under a subscription identifier that the ADRF mapped to that transRefId, and only when it arrives on the notification URI the ADRF gave out.	`network`	proved	a08b	checked
K01	The attacker on the network cannot compute the access token credential a requester uses with the ADRF for the storage scope.	`network`	proved	k01	none
Z01	The ADRF accepts a storage subscription creation only with a token whose subject is the NF that sent the request on its own TLS identity, with the ADRF as audience and the storage scope.	`network`	proved	z01	checked

target	claim	attacker	answer	query	control
Z01	The NWDAF accepts an upstream subscription only with a token whose subject is the ADRF that sent it, with the NWDAF as audience and the event subscription scope.	`network`	proved	z01n	checked
Z03	When the ADRF removes the upstream subscription for a transRefId, the removal request carried a valid token of the subject that owns that transRefId: no other NF can have the ADRF remove another consumer's storage subscription.	`network`	proved	z03	checked
Z03	When the ADRF removes the upstream subscription for a transRefId, the removal request is bound to the subject that owns that transRefId, so another authorized NF that knows the identifier cannot have it removed.	`leaked_transref`	attack (holds_in_source)	z03g	
S05	After the ADRF has accepted the removal for a transRefId, a further notification on that subscription is not stored: the removed transRefId accepts no new stored notification.	`network`	bounded (the changing value unrolled to 3 steps)	s05	checked
S05	A transRefId whose removal was accepted creates no new upstream subscription afterwards, and a repeated removal of the same transRefId is not accepted a second time: the one-time action happens once.	`network`	bounded (the changing value unrolled to 3 steps)	s05b	checked
A08.8	The record the ADRF stores is filed under the transRefId and the upstream subscription identifier the ADRF itself mapped to that transRefId, not under a subscription of another request.	`network`	proved	a08b	checked
U01	The ADRF creates a storage subscription only on a request sent by that consumer itself, with that consumer's target, event and data set tag; no other consumer causes it.	`network`	proved	u01	checked
S05.10	The storage subscription removal removes the upstream subscription only: the data or analytics already stored for the transRefId are not deleted by it, and no deletion of a stored record happens on that path.	`network`	bounded (the changing value unrolled to 3 steps)	s0510	checked
X01	The analytics carried in a notification the ADRF stores, and the parameters of the storage subscription and removal requests, are never obtainable or alterable by an attacker on the network: every message travels only under the endpoint TLS keys.	`network`	proved	x01	none

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-20261115T092715, 2026-11-15. Status: finished. Model: deepseek-v4-1-flash-261017. Steps: 65, prover 4 s, wall 17 min, cost 0.120 USD.

ProVerif: 95 proofs run, 95 decided, 0 without an answer, 3 s, 25 answers recorded.

What was analysed

Protocol	ADRF subscribe and store analytics
Version	19.7.0
Analysed	One requester, ADRF, NWDAF analytics producer; aggregated non-UE analytics, StorageSubscriptionRequest triggers one upstream NWDAF subscription, the notification is stored in the ADRF, then StorageSubscriptionRemoval. Includes the real upstream API and ID mapping; excludes the DCCF/MFAF/UDM consent branch.
Not analysed	The symbolic model does not prove the computational security or post-quantum resistance of the algorithms; attack traces hold only under explicit assumptions and are not directly equivalent to specification vulnerabilities. Every property must have a reachability/positive control. Rel-19 is fixed. The external network attacker is distinguished from a malicious legitimate NF holding its own credentials; TLS authenticated encryption is modelled per endpoint, and a legitimate NF cannot read another NF's TLS session. OAuth2 subject/audience/scope/expiry and TLS identity binding follow the attached baseline; a service scope does not automatically equal resource ownership. Monotonic tables cannot simulate real deletion, and the finite-state lifecycle must be explicitly bounded. The target is the specified API sub-flow, not complete 5G security.
Source text	1182 numbered passages (839894 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	4.2.2.3	Nadrf_DataManagement_StorageSubscriptionRequest service operation
main	4.2.2.3.1	General
main	4.2.2.3.2	Requesting subscription and storage of data or analytics
main	4.2.2.4	Nadrf_DataManagement_StorageSubscriptionRemoval service operation
main	4.2.2.4.1	General
main	4.2.2.4.2	Requesting removal of subscription of data or analytics
main	5.1	Nadrf_DataManagement Service API
main	5.1.1	Introduction
main	5.1.2	Usage of HTTP
main	5.1.2.1	General
main	5.1.2.2	HTTP standard headers
main	5.1.2.2.1	General
main	5.1.2.2.2	Content type

Document	Section	Title
main	5.1.2.3	HTTP custom headers
main	5.1.3	Resources
main	5.1.3.1	Overview
main	5.1.3.2	Resource: ADRF Data Store Records
main	5.1.3.2.1	Description
main	5.1.3.2.2	Resource Definition
main	5.1.3.2.3	Resource Standard Methods
main	5.1.3.2.3.1	POST
main	5.1.3.2.3.2	GET
main	5.1.3.2.4	Resource Custom Operations
main	5.1.3.3	Resource: Individual ADRF Data Store Record
main	5.1.3.3.1	Description
main	5.1.3.3.2	Resource Definition
main	5.1.3.3.3	Resource Standard Methods
main	5.1.3.3.3.1	DELETE
main	5.1.3.3.4	Resource Custom Operations
main	5.1.3.4	Resource: ADRF Data Retrieval Subscriptions
main	5.1.3.4.1	Description
main	5.1.3.4.2	Resource Definition
main	5.1.3.4.3	Resource Standard Methods
main	5.1.3.4.3.1	POST
main	5.1.3.4.4	Resource Custom Operations
main	5.1.3.5	Resource: Individual ADRF Data Retrieval Subscription
main	5.1.3.5.1	Description
main	5.1.3.5.2	Resource Definition
main	5.1.3.5.3	Resource Standard Methods
main	5.1.3.5.3.1	DELETE
main	5.1.3.5.4	Resource Custom Operations
main	5.1.4	Custom Operations without associated resources
main	5.1.4.1	Overview
main	5.1.4.2	Operation: request-storage-sub
main	5.1.4.2.1	Description
main	5.1.4.2.2	Operation Definition
main	5.1.4.3	Operation: request-storage-sub-removal
main	5.1.4.3.1	Description
main	5.1.4.3.2	Operation Definition

Document	Section	Title
main	5.1.4.4	Operation: remove-stored-data-analytics
main	5.1.4.4.1	Description
main	5.1.4.4.2	Operation Definition
main	5.1.5	Notifications
main	5.1.5.1	General
main	5.1.5.2	Retrieval Notification
main	5.1.5.2.1	Description
main	5.1.5.2.2	Target URI
main	5.1.5.2.3	Standard Methods
main	5.1.5.2.3.1	POST
main	5.1.5.3	ADRF Alert Notification
main	5.1.5.3.1	Description
main	5.1.5.3.2	Target URI
main	5.1.5.3.3	Standard Methods
main	5.1.5.3.3.1	POST
main	5.1.6	Data Model
main	5.1.6.1	General
main	5.1.6.2	Structured data types
main	5.1.6.2.1	Introduction
main	5.1.6.2.2	Type: NadrfDataStoreRecord
main	5.1.6.2.3	Type: NadrfDataStoreSubscription
main	5.1.6.2.4	Type: NadrfDataRetrievalSubscription
main	5.1.6.2.5	Type: NadrfDataRetrievalNotification
main	5.1.6.2.6	Type: NadrfDataStoreSubscriptionRef
main	5.1.6.2.7	Type: NadrfStoredDataSpec
main	5.1.6.2.8	Type: DataSubscription
main	5.1.6.2.10	Type: StorageHandlingInfo
main	5.1.6.2.11	Type: NadrfAlertNotification
main	5.1.6.2.12	Type: NadrfAlertNotificationResponse
main	5.1.6.2.13	Type: DataSetTag
main	5.1.6.3	Simple data types and enumerations
main	5.1.6.4	Data types describing alternative data types or combinations of data types
main	5.1.7	Error Handling
main	5.1.7.1	General
main	5.1.7.2	Protocol Errors
main	5.1.7.3	Application Errors

Document	Section	Title
main	5.1.8	Feature negotiation
main	5.1.9	Security
main	5.1.6.2.9	
ts-29-520	4.2.1	Service Description
ts-29-520	4.2.1.1	Overview
ts-29-520	4.2.1.2	Service Architecture
ts-29-520	4.2.1.3	Network Functions
ts-29-520	4.2.1.3.1	Network Data Analytics Function (NWDAF)
ts-29-520	4.2.1.3.2	NF Service Consumers
ts-29-520	4.2.2.2	Nnwdaf_EventsSubscription_Subscribe service operation
ts-29-520	4.2.2.2.1	General
ts-29-520	4.2.2.2.2	Subscription for event notifications
ts-29-520	4.2.2.2.3	Update subscription for event notifications
ts-29-520	4.2.2.3	Nnwdaf_EventsSubscription_Unsubscribe service operation
ts-29-520	4.2.2.3.1	General
ts-29-520	4.2.2.3.2	Unsubscribe from event notifications
ts-29-520	4.2.2.4	Nnwdaf_EventsSubscription_Notify service operation
ts-29-520	4.2.2.4.1	General
ts-29-520	4.2.2.4.2	Notification about subscribed event
ts-29-520	5.1.1	Introduction
ts-29-520	5.1.2	Usage of HTTP
ts-29-520	5.1.2.1	General
ts-29-520	5.1.2.2	HTTP standard headers
ts-29-520	5.1.2.2.1	General
ts-29-520	5.1.2.2.2	Content type
ts-29-520	5.1.2.3	HTTP custom headers
ts-29-520	5.1.3.1	Resource Structure
ts-29-520	5.1.3.2	Resource: NWDAF Events Subscriptions
ts-29-520	5.1.3.2.1	Description
ts-29-520	5.1.3.2.2	Resource definition
ts-29-520	5.1.3.2.3	Resource Standard Methods
ts-29-520	5.1.3.2.3.1	POST
ts-29-520	5.1.3.2.4	Resource Custom Operations
ts-29-520	5.1.3.3	Resource: Individual NWDAF Event Subscription
ts-29-520	5.1.3.3.1	Description
ts-29-520	5.1.3.3.2	Resource definition

Document	Section	Title
ts-29-520	5.1.3.3.3	Resource Standard Methods
ts-29-520	5.1.3.3.3.1	DELETE
ts-29-520	5.1.3.3.3.2	PUT
ts-29-520	5.1.3.3.4	Resource Custom Operations
ts-29-520	5.1.5	Notifications
ts-29-520	5.1.5.1	General
ts-29-520	5.1.5.2	Event Notification
ts-29-520	5.1.5.2.1	Description
ts-29-520	5.1.5.2.2	Operation Definition
ts-29-520	5.1.6.2.2	Type NnwdafEventsSubscription
ts-29-520	5.1.6.2.3	Type EventSubscription
ts-29-520	5.1.6.2.4	Type NnwdafEventsSubscriptionNotification
ts-29-520	5.1.6.2.5	Type EventNotification
ts-29-520	5.1.6.2.7	Type EventReportingRequirement
ts-29-520	5.1.6.2.22	Type NetworkPerfRequirement
ts-29-520	5.1.6.2.23	Type NetworkPerfInfo
ts-29-520	5.1.6.2.30	Type ThresholdLevel
ts-29-520	5.1.6.2.49	Type ConsumerNfInformation
ts-29-520	5.1.6.3.3	Enumeration: NotificationMethod
ts-29-520	5.1.6.3.4	Enumeration: NwdafEvent
ts-29-520	5.1.6.3.10	Enumeration: NetworkPerfType
ts-29-520	5.1.6.3.26	Enumeration: TermCause
ts-29-520	5.1.7	Error handling
ts-29-520	5.1.7.1	General
ts-29-520	5.1.7.2	Protocol Errors
ts-29-520	5.1.7.3	Application Errors
ts-29-520	5.1.8	Feature negotiation
ts-29-520	5.1.9	Security

The purpose of the protocol as the source step located it in the documents: a requester, the ADRF, and an NWDAF analytics producer; it aggregates non-UE analytics, StorageSubscriptionRequest triggers an upstream NWDAF subscription, the notifications are stored in the ADRF, and then StorageSubscriptionRemoval follows. It covers the real upstream API and ID mapping; it excludes the DCCF/MFAF/UDM consent branches.

The properties against the documents

Before writing a query, the analyst checked 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 do exist in the documents.

ID	Name	About	What the documents say	Where
A02	upstream peer authentication at ADRF	The ADRF is assured that a subscription response or event notification it accepts comes from a real upstream NWDAF whose intended peer is the ADRF.	The ADRF subscribes to the NWDAF (s0026) and the NWDAF notifies the notificationURI it received (s0913); NF service procedures shall support mutual authentication (ts-33-501 P219); API access is protected by TLS and OAuth2 (s0334, s1138).	s0026, s0334, s0913, s0961, s1138
A03	agreement on upstream subscription parameters	The ADRF and the upstream NWDAF agree on the subscription created: the ADRF's identity, the subscribed event and the notification URI.	The ADRF creates the upstream subscription according to the consumer's parameters (s0026, s0018) and the NWDAF answers 201 Created with the created subscription and Location (s0855).	s0008, s0021, s0026, s0287, s0855
A08	stored record bound to requester request	The ADRF stores a notification only under the subscription (transRefId) for which it created the upstream subscription, with the data set tag the requester gave, and only when the notification arrives on the notification URI the ADRF gave out.	The ADRF assigns the transaction reference identifier (s0021) and maps it to the upstream subscription (s0026); the data set tag is associated with the subscription and the collected data (s0020, s0313).	s0020, s0021, s0026, s0313
K01	OAuth2 token secrecy	The access token a requester, the ADRF or the NWDAF holds for the Nadrf_DataManagement and Nnwdaf_EventsSubscription APIs is not learnable by the attacker.	Tokens are obtained from the NRF and used to authorize API access (s0335, s0337, s1139); access token credentials must be kept confidential (rfc6749 P011).	s0334, s0335, s0337, s1139
Z01	authorization of API operations	The ADRF accepts a storage subscription creation or removal only with a token whose subject is the requesting NF, audience the ADRF and scope the storage scope; the NWDAF accepts an upstream subscription only with a token for itself.	Access to the APIs may be authorized with OAuth2 client credentials and the scopes nadrf-datamanagement:storage-read-delete-subs and nnwdaf-eventssubscription (s0334, s0339, s1143).	s0334, s0337, s0339, s1143
Z03	removal bound to owning subject	When the ADRF removes an upstream subscription for a transRefId, that transRefId belongs to the subject of the token of the removal request.	The removal request names a transRefId or dataSetId (s0034) and the ADRF removes the subscription mapped to it (s0035); the documents state no rule binding the identifier to the requesting NF.	s0034, s0035, s0293
S05	lifecycle after removal	After the ADRF accepted the removal, the transRefId no longer creates upstream subscriptions nor accepts stored notifications, and an upstream subscription still referenced by another active transRefId is not deleted.	The removal removes the subscription mapped to the transaction reference identifier unless further active transaction reference identifiers are mapped to it (s0035); stored data are removed by a separate operation (s0029, s0215).	s0029, s0035, s0186, s0215
A08.8	record identity correspondence	The record the ADRF stores carries the transRefId and data set tag of the subscription the requester created and the notification URI the ADRF itself gave to the NWDAF.	The ADRF maps the assigned transaction reference identifier to the subscription serving the transaction (s0026) and stores notified data or analytics (s0029, s0035); the parameters from the consumer are used for the upstream subscription (s0018).	s0018, s0021, s0026, s0035
U01	no cross-consumer parameter change	A malicious NF service consumer B cannot cause the ADRF to create or change the collection or storage parameters of a subscription of consumer A.	The storage subscription carries the consumer's subscription, target and storage handling parameters (s0008, s0017, s0019, s0020) and the ADRF uses them for the upstream subscription (s0026); the documents do not state who may change them.	s0008, s0017, s0019, s0020, s0026

ID	Name	About	What the documents say	Where
S05.10	removal does not delete stored records	StorageSubscriptionRemoval stops collection and removes the upstream subscription, while already stored data or analytics are only removed by remove-stored-data-analytics.	The removal operation removes the subscription (s0035); a separate operation removes already stored data or analytics (s0186, s0215).	s0029, s0035, s0186, s0215
X01	confidentiality and integrity of API messages in transit	The analytics the upstream NWDAF delivers to the ADRF for storage, and the parameters of the storage subscription and removal requests	The NWDAF sends the notification to the notificationURI over the service based interface (s0913); NF service procedures shall support confidentiality and integrity (ts-33-501 P217).	s0913, s0961, s0334

Added from the documents by the analyst: X01.

The security statements of the documents, paragraph by paragraph

The host listed the 398 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: 42. Not taken: 356. 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 P002	rfc6749:2778-2786	No goal stated	Introduction; states no requirement on this protocol.
P003 to P006	rfc6749:2790-2823	`compromised_consumer`	
P007 to P010	rfc6749:2827-2850	Z01, `compromised_consumer`	
P011 to P015	rfc6749:2859-2881	K01, `compromised_consumer`	
P016 to P021	rfc6749:2885-2917	Outside the scope	Refresh tokens belong to other grants; this protocol uses the client credentials grant.
P022 to P029	rfc6749:2921-2980	Outside the scope	Authorisation codes and redirection URIs are not used with the client credentials grant.
P030 to P033	rfc6749:2984-3003	Outside the scope	The resource owner password grant is not used.
P034 to P035	rfc6749:3013-3019	K01	
P036	rfc6749:3023-3028	A02, Z01	
P037 to P039	rfc6749:3032-3041	The prover cannot be asked	Guessing of client credentials: the strength of a secret cannot be asked; this protocol uses NF credentials, not user passwords.
P040 to P050	rfc6749:3045-3138	Outside the scope	Phishing, CSRF and clickjacking concern user agents; this protocol has none.
P051 to P052	rfc6749:3142-3150	No goal stated	Advice to implementers on input validation; no property of this message flow.
P053 to P054	rfc6749:3154-3167	Outside the scope	Open redirectors; no redirection endpoint here.
P055 to P062	rfc6749:3170-3210	Outside the scope	The implicit grant is not used.

Paragraphs	Where	Taken up as	Why not
P063	ts-33-501:2491-2491	The prover cannot be asked	Algorithm bidding down: the strength of algorithms is outside what the prover can be asked.
P064 to P072	ts-33-501:2495-2511	Z01, K01, `network`, `compromised_consumer`	
P073 to P074	ts-33-501:2515-2517	Outside the scope	Keys of the 5G core and NG-RAN; this protocol uses none.
P075 to P213	ts-33-501:2523-2853	Outside the scope	User data, signalling data, credentials and subscriber privacy on the access and core interfaces; the task covers the ADRF API sub-flow only.
P214	ts-33-501:2859-2859	No goal stated	It states the trust-zone assumption the requirements rest on.
P215 to P220	ts-33-501:2865-2875	A02, Z01, `compromised_consumer`	
P221 to P227	ts-33-501:2879-2891	Z01, K01, `compromised_consumer`	
P228 to P241	ts-33-501:2895-2923	Outside the scope	NEF and SCP requirements; the NEF is excluded and direct communication is modelled.
P242 to P314	ts-33-501:2929-3081	Outside the scope	Roaming, SEPP, PRINS, IPUPS and attribute protection between networks; one network without roaming is analysed.
P315 to P318	ts-33-501:3085-3091	A02, `compromised_consumer`	
P319 to P337	ts-33-501:3097-3141	No goal stated	Signalling monitoring, visibility and configurability: operational requirements without a property of this message flow.
P338 to P375	ts-33-501:3149-3231	The prover cannot be asked	Algorithm identifiers and selection: the strength of ciphers is outside what the prover can be asked.
P376 to P392	ts-33-501:6540-10909	Outside the scope	N3, EAP-TLS attachment and IAB requirements; not part of this API sub-flow.
P393 to P398	ts-33-501:10913-11901	Outside the scope	IAB secure environment, MBSF and MBSTF requirements and NEF security; not part of this API sub-flow.

The parties and the protected values, one by one

The host listed the 24 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: 17. Without: 7. 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	NF service consumer	party	`compromised_consumer`	
S02	ADRF	party	`leaked_transref`	
S03	Upstream NWDAF	party	`compromised_nwdaf`	
S04	NRF	party	Everything lost by definition	Every token and NF credential of this protocol comes from the authorisation server; with the NRF controlled the authorisation properties are lost by definition and the documents ask none of them under it.
S05	Requester A	party	`compromised_consumer`	
S06	Requester B	party	`compromised_consumer`	

ID	Party or value	Kind	Attacker	Why none
S07	Upstream subscription	value	`compromised_nwdaf`	
S08	Notification content	value	`compromised_nwdaf`	
S09	Subscription parameters	value	`compromised_consumer`	
S10	Mapping between transRefId and the upstream subscription	value	`leaked_transref`	
S11	targetNfId/targetNfSetId	value	Nothing gained	targetNfId is a public NF instance identifier discovered from the NRF; it is not a secret.
S12	dataSetTag/dataSetId	value	`leaked_transref`	
S13	notifCorrId	value	Nothing gained	notifCorrId is a correlation identifier the consumer chooses; s0287 requires uniqueness per consumer, not secrecy.
S14	transRefId	value	`leaked_transref`	
S15	OAuth2 access token	value	Everything lost by definition	The holder of a token is the party it was issued to; asking that it cannot use its own token asks nothing.
S16	NF credentials	value	`compromised_consumer`	
S17	Stored subscription	value	`leaked_transref`	
S18	Upstream subscription	value	`compromised_nwdaf`	
S19	Stored record	value	Outside the scope	Reading stored records is the retrieval sub-flow, which the task leaves out.
S20	dataSetId	value	`leaked_transref`	
S21	Stored records	value	Outside the scope	Stored records and their removal belong to the retrieval and remove-stored-data-analytics sub-flows, which the task leaves out.
S22	Upstream subscription parameters	value	`compromised_consumer`	
S23	Storage handling parameters	value	`compromised_consumer`	
S24	dataSetTag	value	Nothing gained	dataSetTag is a label the consumer chooses for its own subscription (s0020, s0313); the tag is not a secret, only the identifier binding is answered under leaked_transref.

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		
`compromised_consumer`	the credentials and stored state of an authorized NF service consumer of the ADRF API	The consumer holds an OAuth2 token and TLS credentials (s0334, s0335); a compromised consumer is the malicious legal NF named in the scope	s0334, s0335

Attacker	Has	Basis	Where
`compromised_nwdaf`	the credentials and subscription state of a real upstream NWDAF analytics producer	The NWDAF authenticates itself with OAuth2 and TLS to the ADRF (s1138, s1139)	s1138, s1139
`leaked_transref`	the transaction reference identifier the ADRF assigned to another consumer's storage subscription	s0293 defines transRefId as a plain string attribute and no clause states a confidentiality rule for it	s0021, s0034, s0293

ID	`network`	`compromised_consumer`	`compromised_nwdaf`	`leaked_transref`
A02	Holds			
A03	Holds			
A08	Holds			
K01	Holds			
Z01	Holds			
Z03	Holds			Fails
S05	Holds			
A08.8	Holds			
U01	Holds			
S05.10	Holds			
X01	Holds			

Source and targets

Source: 1182 units, 839894 bytes, frozen in source-delivery-01.json. Targets: 8 approved (A02, A03, A08, K01, Z01, Z03, S05, 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 the source package protocol-97d3615022da9a42cbd490b5b948

This step as it reports itself: 2 requests (2 failed), 0.000 USD, 718123 tokens in and 9369 out, 294 s.

Notes on the prover and the model

- Z03: answered in ProVerif, and the analyst names changing_state for this target
- S05: answered in ProVerif, and the analyst names changing_state for this target
- S05.10: answered in ProVerif, and the analyst names changing_state 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.

A bound in steps is an answer on bounded.pv: the analyst's second ProVerif model, in which the party that keeps a value that changes is unrolled to that many steps with its check in place. That model does less than the protocol does, so the property holds for those steps and no further, and an attack found there is an attack. The number of steps is the analyst's statement.

State review by the analyst

This part is the analyst's own reading of the source, not an answer of the prover. "yes" under shown by ProVerif means that ProVerif found a trace with these steps in the unrolled model, so the steps are possible; that they break a target is said only by an attack in the table of answers, and what else they cost is the analyst's reading. "nothing found" says which patterns were looked at and proves nothing.

value	kept by	changes when	checked when	refs
the transRefId together with the upstream subscription it is mapped to	the ADRF	on acceptance of a StorageSubscriptionRequest the ADRF assigns a new transRefId and records the mapping to the upstream subscription it creates; an accepted removal clears what the transRefId does (s0021, s0026, s0035)	when a StorageSubscriptionRemoval names it, and when a notification has to be filed under it (s0034, s0035)	s0021, s0026, s0034, s0035, s0036
the upstream subscription identifier subscriptionId and its notification URI	the upstream NWDAF	the NWDAF creates it on 201 Created and deletes it on an accepted HTTP DELETE (s0855, s0898)	on every HTTP DELETE naming the subscriptionId and on every notification for that subscription (s0897, s0913, s0956)	s0855, s0897, s0898, s0901
the stored record with its data set tag and its lifetime	the ADRF	the ADRF stops filing data under the transRefId on an accepted removal; a stored record is only removed by remove-stored-data-analytics (s0035, s0215)	when a notification is to be stored for the subscription, and when stored data are retrieved or removed (s0961, s0962)	s0029, s0035, s0186, s0215
the storage handling information (storeHandl) and the data set tag of a subscription	the ADRF	the ADRF determines the storage approach on acceptance and may change it in the response (s0024)	when the storage lifetime is applied to the collected data (s0024, s0025)	s0024, s0025

pattern	result	target	shown by ProVerif	steps
replay	nothing found			With the source's rule a notification after the accepted removal is refused (s05 verified) and a repeated removal is consumed without a second removal (s05b verified); the control without the replaced value stores it (ctl_s05 falsified).
repeat_one_time_action	nothing found			A repeated removal reaches the ADRF again but is not acted on and no upstream subscription is deleted twice; the ADRF answers nothing to it, which the source does not regulate.
desynchronisation	candidate		no: not tried with ProVerif	The consumer sends request-storage-sub-removal with its transRefId (s0034). / The ADRF accepts it and answers 204 No Content at once (s0035). / Subsequently the ADRF sends HTTP DELETE for the upstream subscription (s0035, s0897). / If the DELETE fails, the NWDAF answers an HTTP error and keeps the subscription (s0901) and keeps notifying on the subscription (s0913). / The consumer believes the collection has stopped while the NWDAF keeps the subscription; the ADRF refuses the further notifications, so nothing more is stored, but no clause makes it retry the deletion or tell the consumer.
update_before_check	candidate		no: not tried with ProVerif	The consumer sends request-storage-sub with targetNfId and anaSub (s0008). / The ADRF assigns the transRefId and answers 200 OK with it before it creates the upstream subscription (s0021). / Only subsequently does the ADRF create the subscription at the NWDAF (s0026). / If that creation fails, the transRefId exists at the consumer and at the ADRF with no upstream subscription, and the source states no rule for it.

pattern	result	target	shown by ProVerif	steps
no_rollback_on_failure	candidate		no: not tried with ProVerif	The ADRF assigns the transRefId and answers 200 OK (s0021). / The creation of the upstream subscription fails or is abandoned (s0026, s0023); no clause removes the transRefId or tells the consumer. / A removal request naming that transRefId is accepted and answered 204 (s0035) although nothing is mapped to it. / The consumer has no way to learn that no upstream subscription ever existed.
race_between_sessions	candidate		no: not tried with ProVerif	Consumer A sends request-storage-sub for a data set; the ADRF creates the upstream subscription (s0026). / Consumer B sends request-storage-sub for the same data or analytics; the ADRF, per s0022 and s0026, assigns a second transRefId and may not create a second upstream subscription but map both transRefIds to the one that exists. / Consumer A sends the removal for its transRefId; s0035 keeps the upstream subscription because B's transRefId is still mapped to it. / The bounded model has one transRefId per upstream subscription, so this keeping rule and the race between the two removals is not exercised.
distinguishable_failure	candidate		no: not tried with ProVerif	Consumer B sends request-storage-sub-removal naming a guessed transRefId (s0034). / The ADRF answers 204 No Content if a subscription is mapped to it and an HTTP error response otherwise (s0035, s0036). / The two answers tell B whether the guessed transRefId exists, and an existing one is removed regardless of its owner (the Z03 attack).
wrap_or_window	nothing found			transRefId and dataSetId are strings with no range, counter or acceptance window in the source.

What the analyst named for targets:

step	targets	features	reason
40	S05, S05.10, Z03	changing_state	S05, S05.10 and the removal-binding claim of Z03 rest on the ADRF's subscription lifecycle: the transRefId and the upstream subscription mapping are created once, are removed by the removal request, and a notification arriving after the removal must not be stored. model.pv keeps no such state (tables only grow and ProVerif forgets how often a step ran), so these answers come from bounded.pv, where the ADRF is unrolled to a fixed number of steps.
40	A02, A03, A08, A08.8, K01, Z01, U01, X01	plain_primitives, many_sessions	A02, A03, A08, A08.8, K01, Z01, U01 and X01 use only a MAC, an authenticated-encryption abstraction for TLS and a private key function per endpoint identity; the properties are asked for any number of sessions of two consumers, the ADRF and the NWDAF. No arithmetic law is involved, so an attack found would be an attack on the protocol too and a proof is a proof against this calculus.

The analyst's closing note

All eleven targets are answered. Ten hold against an attacker on the network alone, each with a control where a variant exists: peer authentication of the upstream NWDAF (A02), agreement on the upstream subscription parameters (A03), binding of a stored notification to the consumer's request, data set tag and mapped subscription (A08, A08.8), token secrecy (K01), the subject/audience/scope binding of both APIs (Z01), the lifecycle after an accepted removal and the separation of removal from stored records (S05, S05.10, unrolled to three steps), no cross-consumer parameter change (U01) and confidentiality and integrity of the messages in transit (X01). Z03 has both readings: it holds for the network attacker because the transRefId cannot be guessed, and it fails for a consumer that knows another consumer's transRefId, because the ADRF removes the upstream subscription on any valid token and the source states no ownership rule. That attack is the main finding, recorded as holds_in_source; the recommendation (record the owning consumer and compare it with the token subject on removal) was checked on a copy of the unrolled model, where the attack query becomes verified and the other recorded answers still hold. A

second finding is read from the source rather than shown by a query: the storage subscription is confirmed before the upstream subscription exists, with no rollback. What remains unmodelled: the compromised consumer and the compromised producer, the s0035 sharing rule for one upstream subscription mapped to several transRefIds, token expiry, and the excluded DCCF/MFAF/UDM branches; TLS is an idealised key per endpoint pair.