Security analysis of CMS key transport plus PSK

This report in another language: 中文.

rfc8696 · ProVerif+ Agent report

Documents	Using Pre-Shared Key (PSK) in the Cryptographic Message Syntax (CMS) (protocol definition)
Properties examined	13: 5 hold, 8 fail (9 attacks)
Findings	3

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

RFC 8696 lets CMS carry content-encryption keys in a way that also needs a pre-shared key, so that breaking the classic key transport, today or with a quantum computer later, is not enough to read the content. The sender generates a content-encryption key and a key-derivation key, transports the key-derivation key to the recipient with the ordinary CMS key transport inside KeyTransPSKRecipientInfo, mixes it with the PSK through a KDF into a key-encryption key, and wraps the content-encryption key with that. The recipient holds the PSK that pskid names and its own key transport private key.

When the exchange ends the sender knows that the content is protected under a key that also needs the PSK. The recipient knows that the content-encryption key comes from some party that holds the PSK and could address a message to the recipient's public key, and it agrees with that party on the key, the algorithm and the content; it does not learn which party that was.

The parties:

- **Sender**: originator, holds the PSK. Holds: PSK (by pskid), recipient's public key (certificate).
- **Recipient**: one recipient, holds the PSK. Holds: PSK (by pskid), key transport private key.

Figure 1 shows the exchange as the documents specify it.

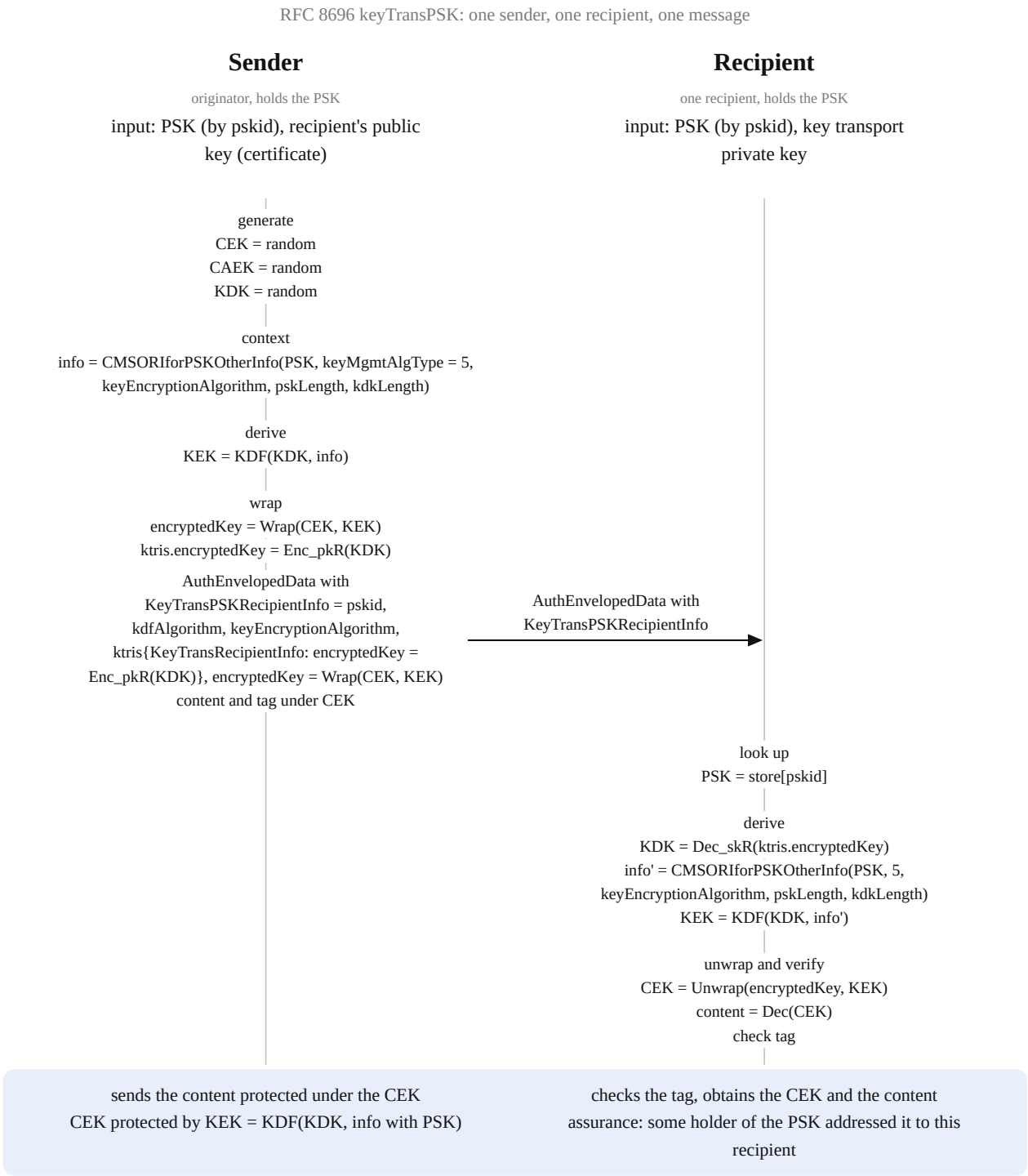


Figure 1. The protocol as specified.

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 3 findings: what follows is not stated in the documents. For 1 of them it is unclear whether the documents state it.

The mechanism gives what it is for: neither the content-encryption key nor the content is reachable by an attacker with the network, and a leaked key-derivation key or a leaked PSK alone is not enough, so the PSK really does carry the confidentiality that the key transport cannot. Three things the documents do not state turned up. The PSK identifier is not part of the derived key, so a message can be re-labelled with another identifier under which the recipient holds the same PSK; any holder of the PSK can put content to the recipient, since the flow names no sender; and a message can be accepted more than once, since nothing in it is fresh. Of the thirteen properties, ten are proved and three are answered by attacks, each proof with a control that fails; the privacy of the PSK identifier fails exactly as the document says. The proofs rest on ideal key transport, KDF and key wrap, on a single recipient, and on lengths being left out of the model.

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 Any holder of the PSK can send content to the recipient as if from the sender

Kind	Weakness of the protocol as specified
New	Unclear: the documents state it in part, or could not be searched (rfc5652:2864, rfc5652:2874, main:649)
Value	medium
Properties	A03, U03 (section 3)
Source text	s0649, s0652
Shown by	ProVerif attack query `a03_forge`; Figures 2 and 3

What happens. KeyTransPSKRecipientInfo names no sender and carries no signature. A party that holds the PSK generates its own key-derivation key, derives the key-encryption key from the PSK, wraps a content-encryption key of its choice to the recipient's public key and sends a complete message, which the recipient accepts exactly as it accepts the real sender's message.

What the attacker needs. The attacker is a member of the group that shares the PSK, or has obtained the PSK; it needs no other secret, since the recipient's public key is public. The document anticipates group PSKs but discusses only eavesdropping.

What the attacker gains. The recipient cannot tell the sender from any other holder of the PSK, so a message the real sender never produced is accepted. Content authenticity in keyTransPSK rests on the PSK alone.

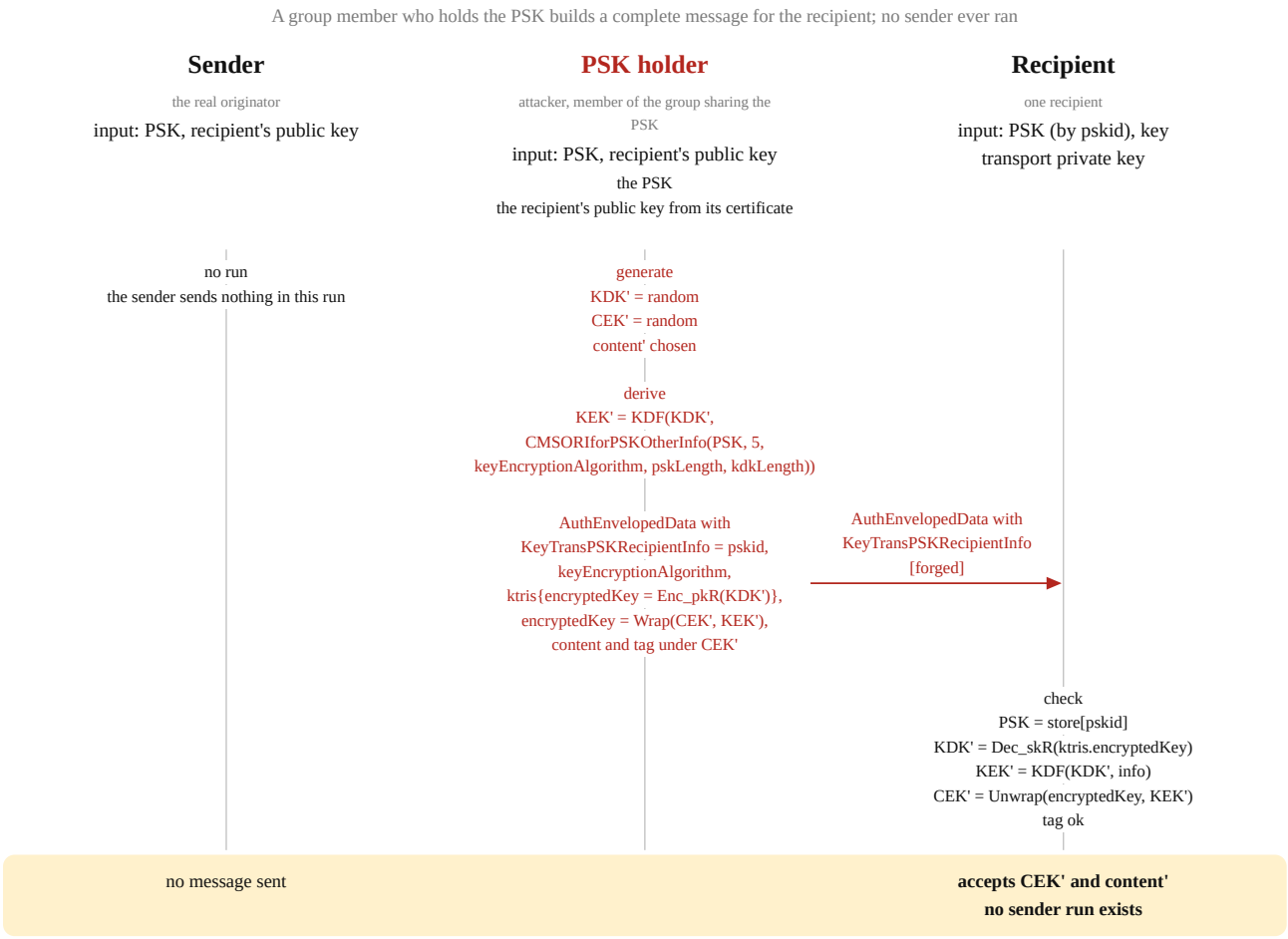


Figure 2. A group member who holds the PSK builds a complete message for the recipient; no sender ever ran. Property A03; attack query a03_forge.

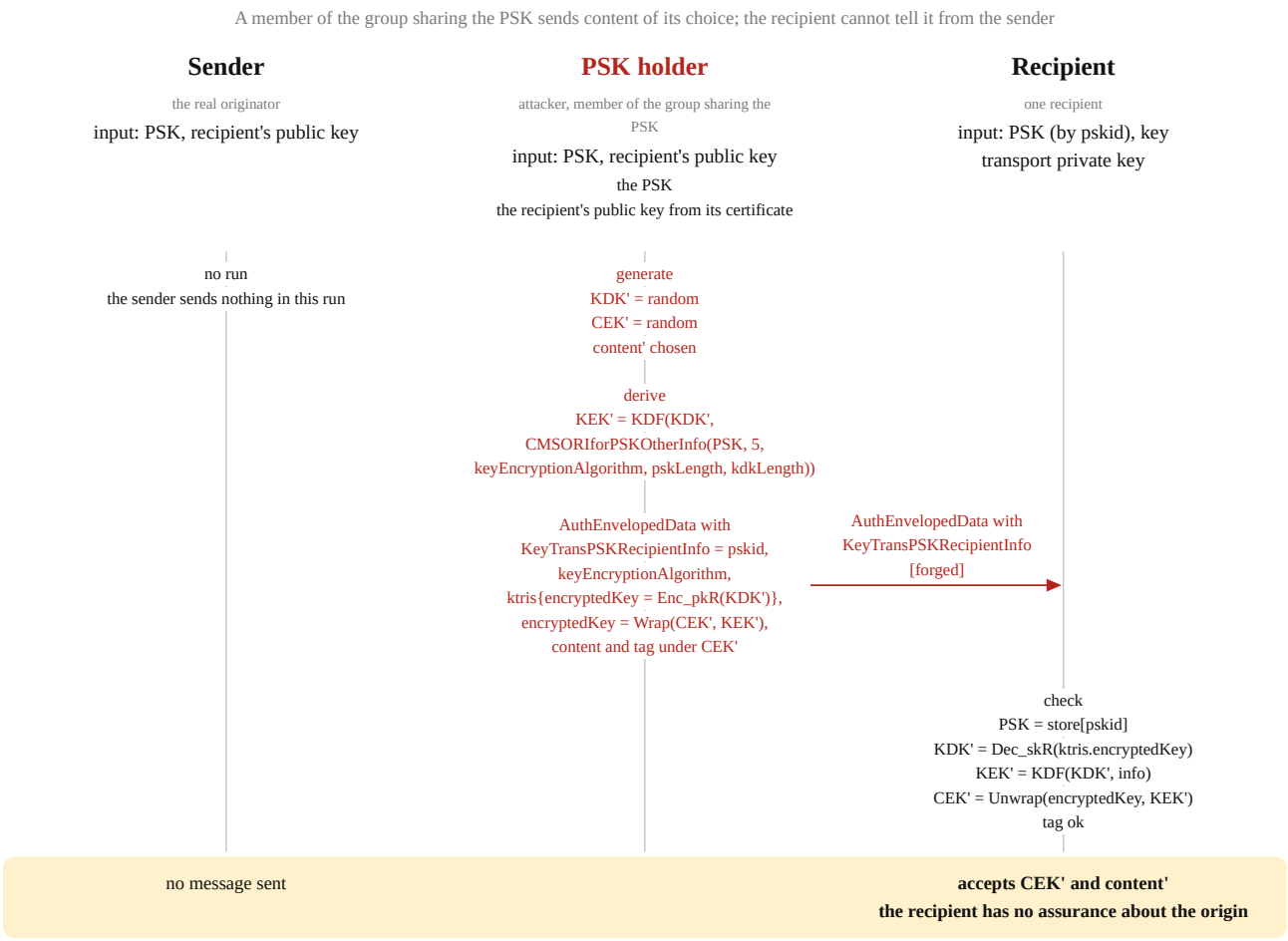


Figure 3. A member of the group sharing the PSK sends content of its choice; the recipient cannot tell it from the sender. Property U03; attack query a03_forge.

Proposed change. The specification should say that the recipient's assurance about the origin is limited to a holder of the PSK, and that a PSK used where the origin matters must be pairwise, one per sender and recipient, rather than shared by a group.

Check of the change with ProVerif. Not checked: A change to pairwise PSKs removes exactly the capability the attacker of this finding has, so the checked model would satisfy the property trivially and say nothing; the change that would keep a group PSK and restore origin assurance is a signature, which the task puts outside the scope and the model therefore does not contain.

2.2 The PSK identifier is outside the derived key, so a message can be re-labelled

Kind	Weakness of the protocol as specified
New	Yes: the documents do not state it
Value	low
Properties	A03, K04, D01, A07, U02 (section 3)
Source text	s0012, s0038, s0039, s0193
Shown by	ProVerif attack query `a03_pskid`; Figures 4, 5, 6, 7 and 8

What happens. The KDF context is the DER encoding of CMSORIforPSKOtherInfo, which carries the PSK value, the key management algorithm type, the key-encryption algorithm and the two lengths, but not the pskid. The recipient resolves the received pskid in its own store, so the key it derives depends on the PSK it finds there, not on the identifier. Where the same PSK is registered under a second identifier, an attacker rewrites pskid to that identifier and the recipient derives the same key-encryption key and accepts.

What the attacker needs. Only the network: the attacker changes the pskid field of a message it has observed. It needs the recipient to hold the same PSK under a second identifier, which the source neither requires nor forbids, since it only calls unique identifiers best practice.

What the attacker gains. Anything an implementation or an application reads from the pskid of an accepted message is unauthenticated: the identifier names a PSK the recipient holds, but not the identifier the sender used. No confidentiality is lost, because the key still depends on the PSK value.

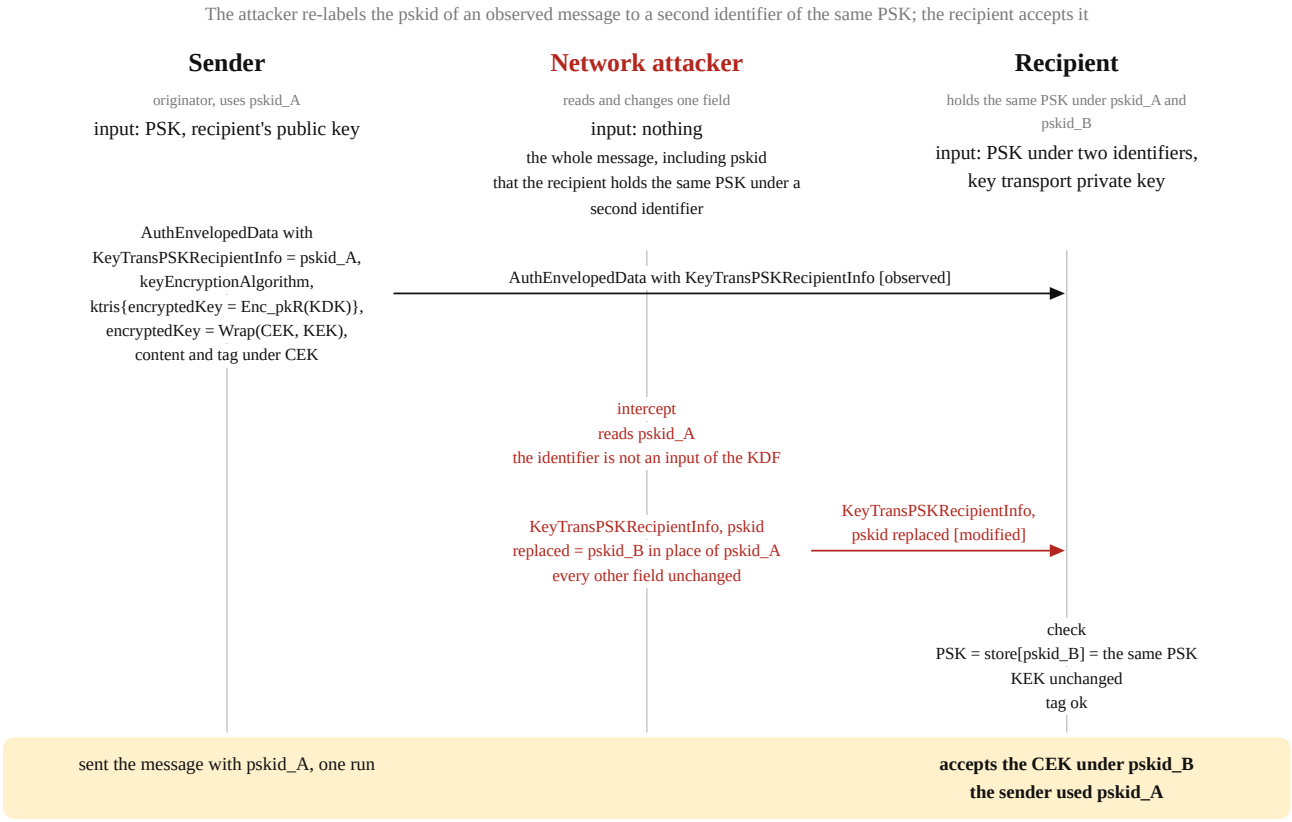


Figure 4. The attacker re-labels the pskid of an observed message to a second identifier of the same PSK; the recipient accepts it. Property A03; attack query a03_pskid.

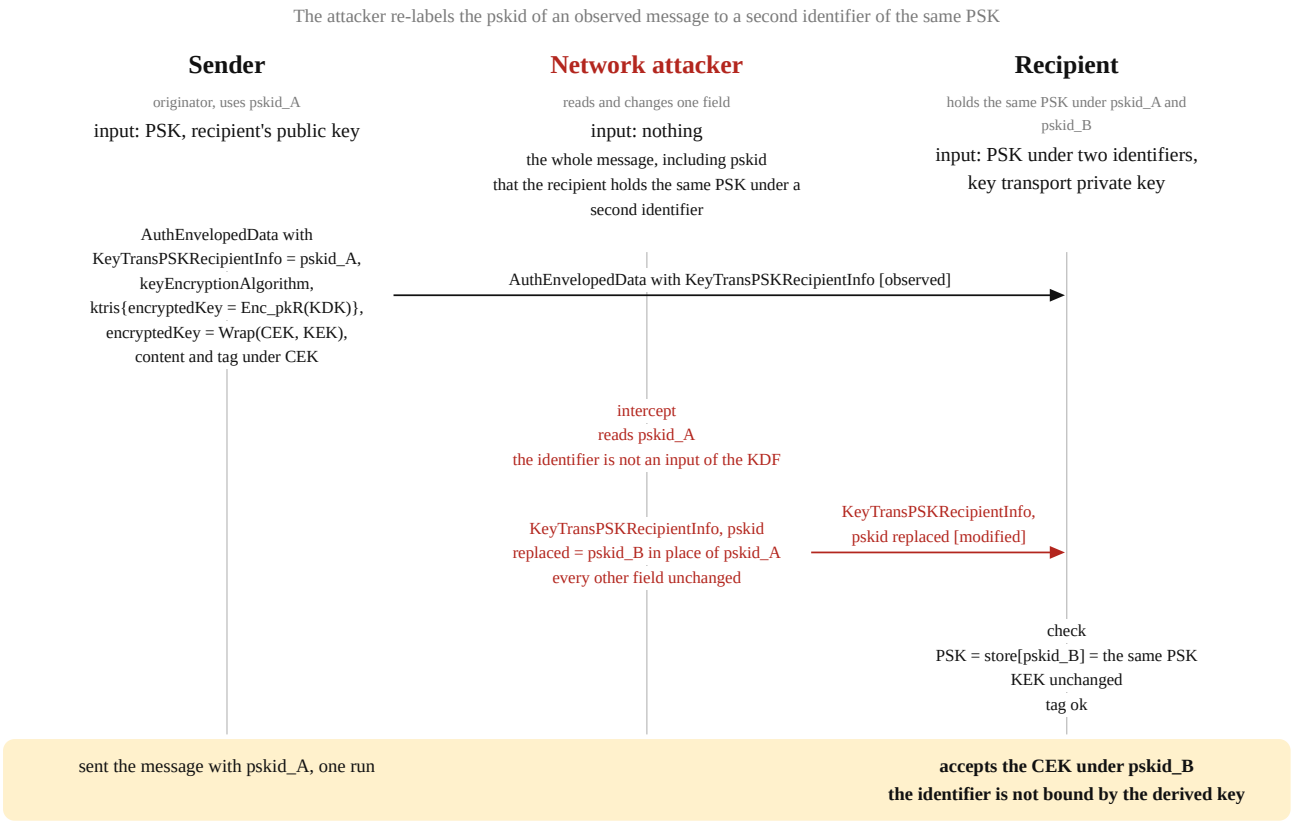


Figure 5. The attacker re-labels the pskid of an observed message to a second identifier of the same PSK. Property K04; attack query a03_pskid.

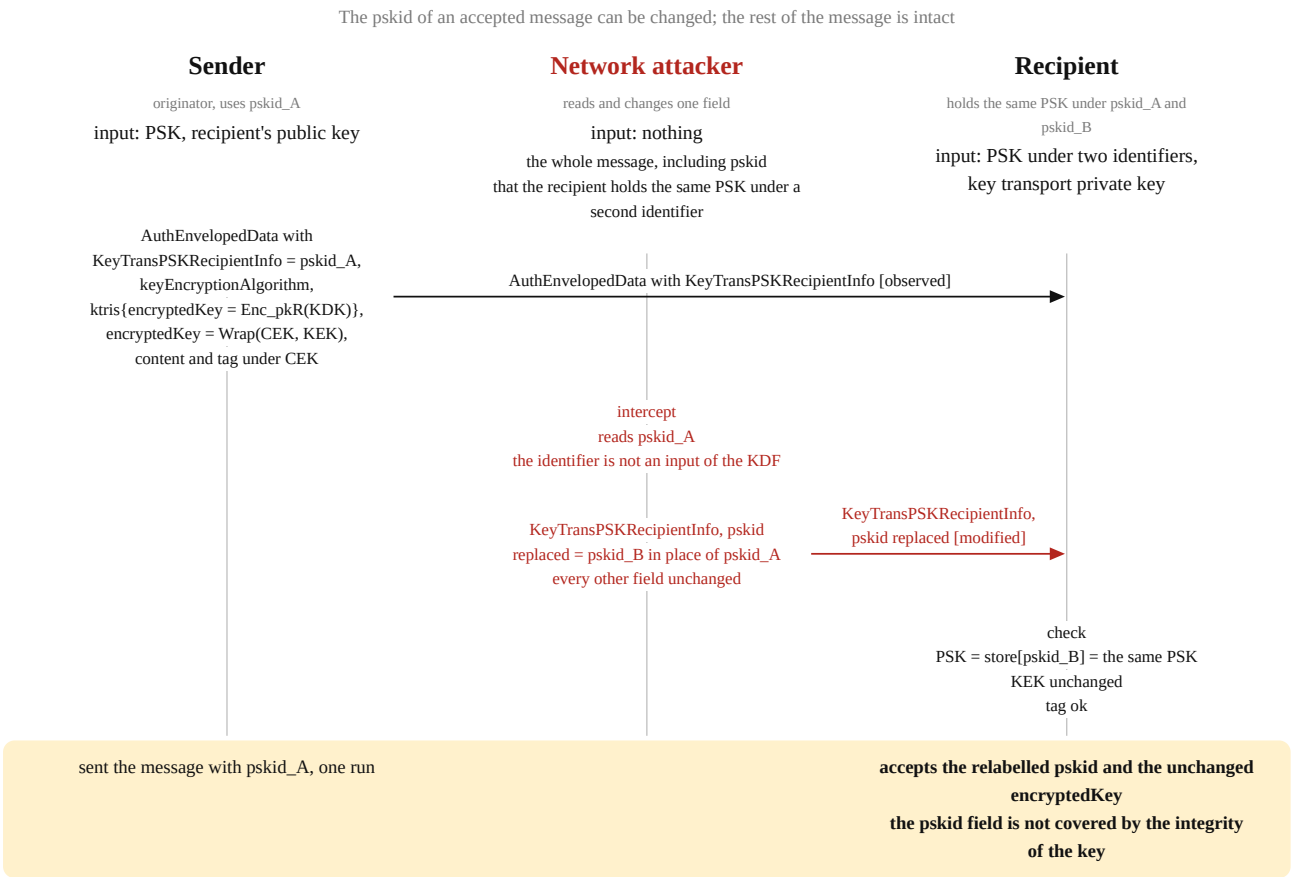


Figure 6. The pskid of an accepted message can be changed; the rest of the message is intact. Property D01; attack query a03_pskid.

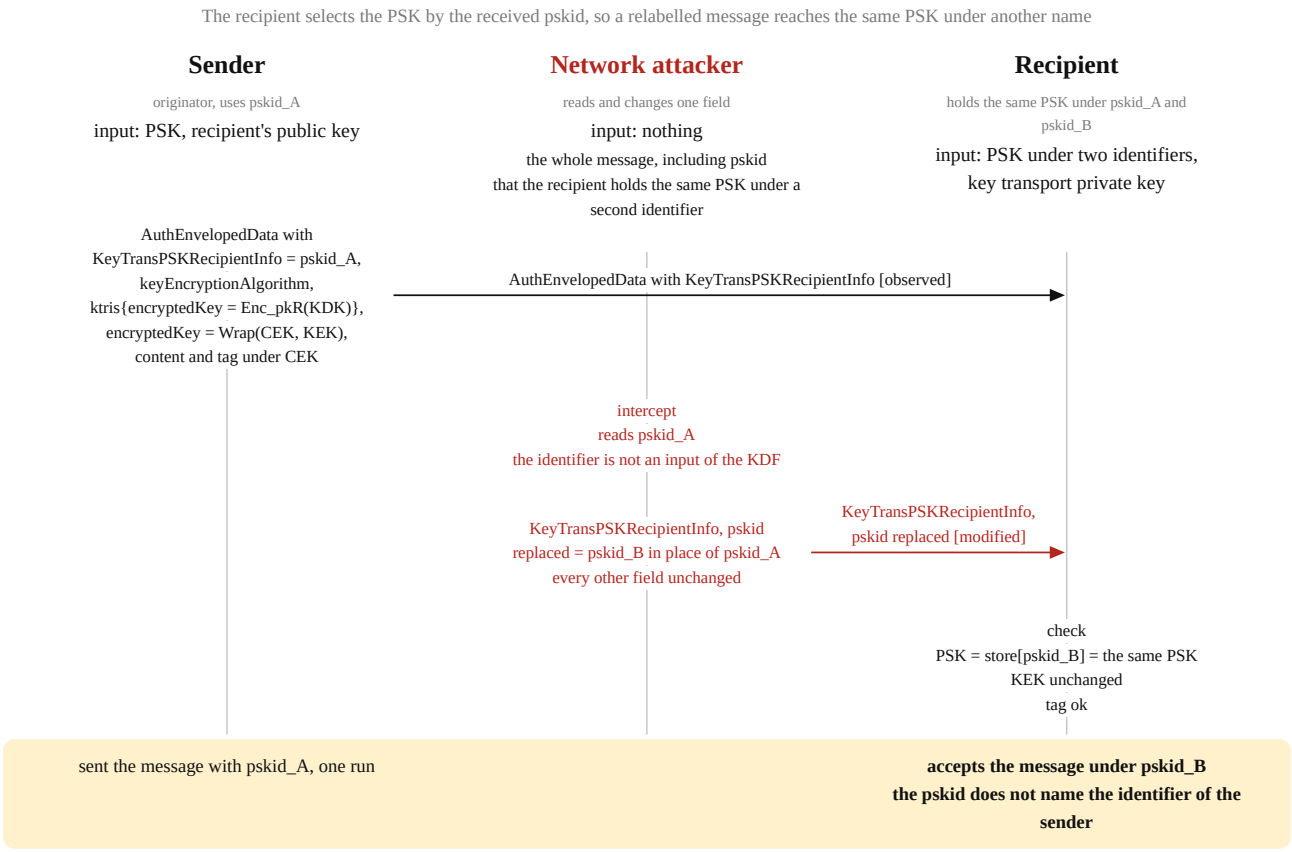


Figure 7. The recipient selects the PSK by the received pskid, so a relabelled message reaches the same PSK under another name. Property A07; attack query a03_pskid.

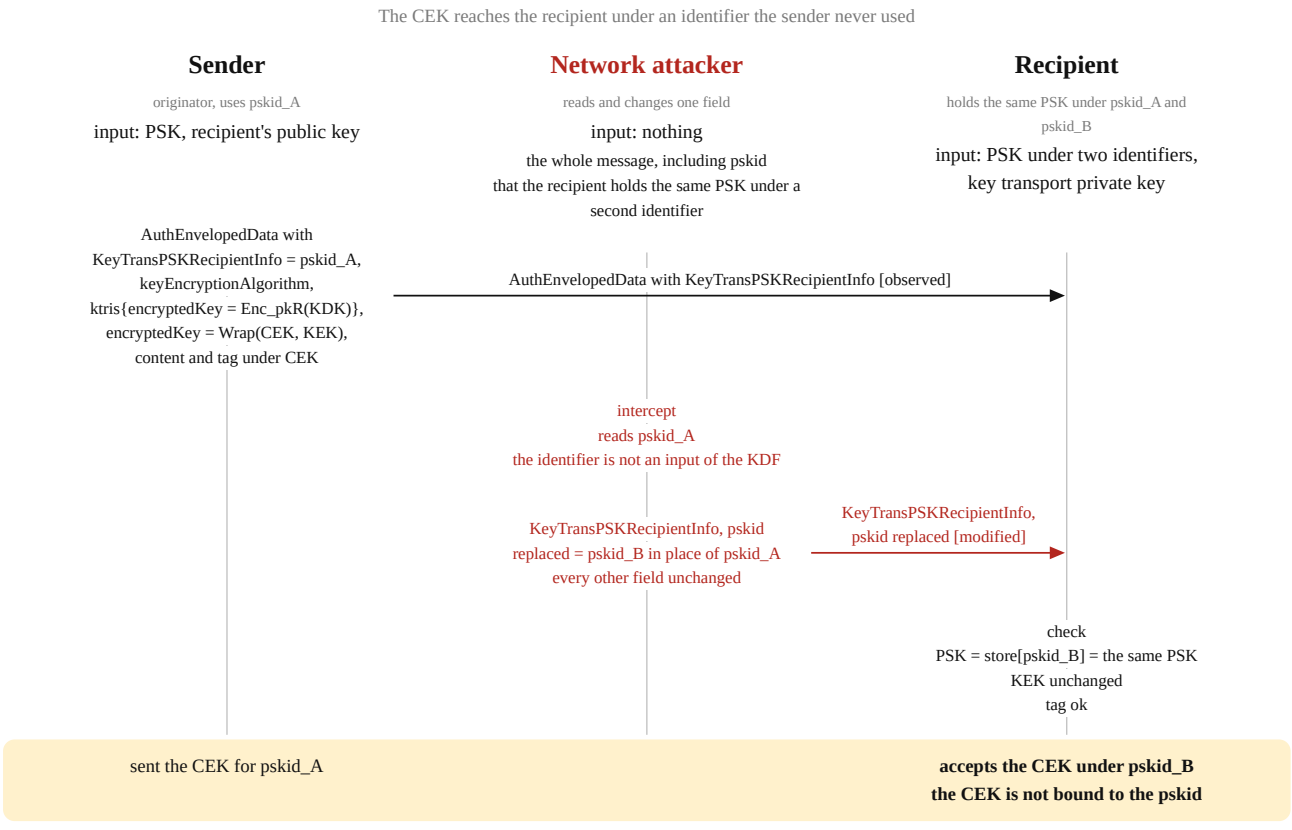


Figure 8. The CEK reaches the recipient under an identifier the sender never used. Property U02; attack query a03_pskid.

Proposed change. Add pskid to CMSORIforPSKOtherInfo, so that the identifier is an input of the KDF and the key-encryption key depends on it as well as on the PSK value.

Check of the change with ProVerif. The change was put into a copy of the model: Add pskid to the KDF context: the CMSORIforPSKOtherInfo input of the KDF carries the PSK identifier, so the derived KEK depends on the pskid as well as on the PSK value. On the changed model ProVerif proves `a03_pskid`: the attack is no longer possible there. The normal run of the protocol is still found. Of the other recorded answers, 21 still hold and 1 changed: `x01_privacy` (X01, was attack) is now inconclusive. 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/0034-fix-pskid.pv`.

2.3 A message instance can be accepted more than once

Kind	Weakness of the protocol as specified
New	Yes: the documents do not state it
Value	low
Properties	A04 (section 3)
Source text	s0002, s0015
Shown by	ProVerif attack query `a04_once`; Figure 9

What happens. The message carries no value that is fresh for the message: no nonce, no counter, no identifier of the instance, and the key-management structure adds none. An attacker on the network delivers the same AuthEnvelopedData message twice and the recipient accepts it twice, on one key-derivation key and one content-encryption key from the sender.

What the attacker needs. Only the network: the attacker records one message and delivers it again.

What the attacker gains. A recipient that treats an accepted content-encryption key as a one-time event, or that counts acceptances, is misled. For plain enveloped content the repeat only re-delivers content the recipient already has.

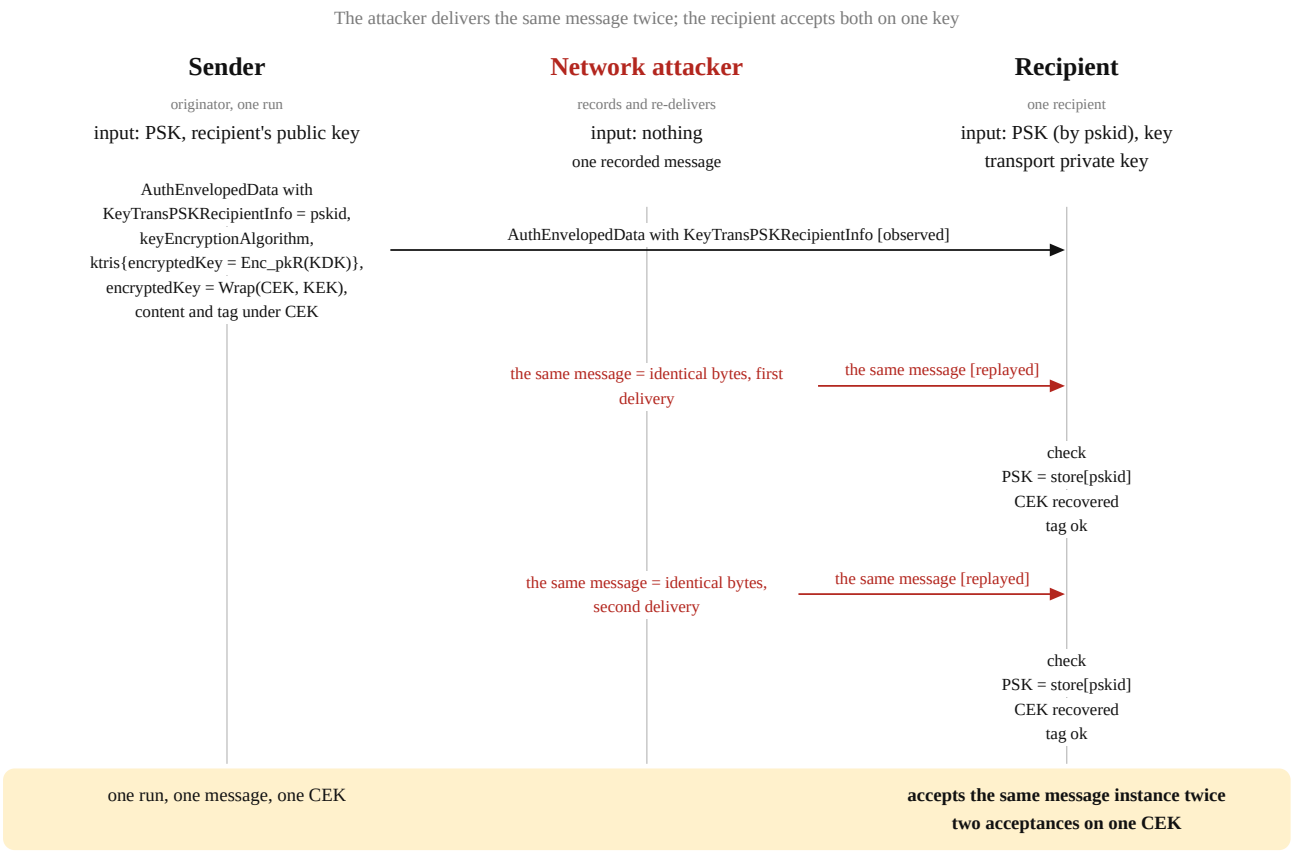


Figure 9. The attacker delivers the same message twice; the recipient accepts both on one key. Property A04; attack query a04_once.

Proposed change. The specification should state whether an accepted content-encryption key may be used once. If it may, the sender must put a fresh value into the KDF context and the recipient must reject a value it has seen, which requires the recipient to keep state.

Check of the change with ProVerif. Not checked: rejecting a repeat needs the recipient to remember earlier messages, and the model of the single-message flow has no such state; a check would need a bounded model with a replay store, which this model cannot express.

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
K01	**Secrecy of CEK, KEK, KDK and PSK.** Underivability is checked separately for CEK/CAEK, KEK, the key-derivation key and the PSK; the legitimate knowers are the sender and the recipient holding that PSK.	no compromise	Holds	medium	yes
K02	**Confidentiality of the protected content.** CEK/CAEK and the content they protect are visible only to the sender and to the recipient holding the same PSK.	no compromise	Holds	medium	yes

ID	Property	Compromise assumed	Result	Value	In the documents
A03	**Agreement on CEK, pskid and algorithm.** Every time the recipient accepts a CEK there is a matching sender run, and the two agree on the identities, the pskid, the keyEncryptionAlgorithm and the CEK.	no compromise	Fails (2 attacks)	medium	unclear
A04	**One acceptance per message instance.** Every acceptance of a CEK by the recipient corresponds to a distinct sender run; the same message instance is accepted at most once.	no compromise	Fails (1 attack)	low	no
K04	**Domain separation of derived KEKs.** KEKs derived under different pskid, keyMgmtAlgType, keyEncryptionAlgorithm or length must be distinct.	no compromise	Fails (1 attack)	low	no
D01	**Integrity of encryptedKey and context.** The encryptedKey and context the recipient accepts correspond to a genuine send by the sender for the pskid, kdfAlgorithm and keyEncryptionAlgorithm, and have not been tampered with.	no compromise	Fails (1 attack)	low	unclear
A07	**pskid selects the recipient's own PSK.** The pskid the recipient accepts matches the identity of the recipient that actually holds that PSK; other recipients' credentials are not mistaken for its own.	no compromise	Fails (1 attack)	low	no
C03	**Session isolation on KDK or PSK leak.** When the key-derivation key or the PSK of one session leaks, the CEKs of other sessions stay secret.	key-derivation key or PSK (after the session completes)	Holds	medium	yes
U01	**CEK secret when the KDK leaks.** When the key-derivation key leaks and the PSK does not, CEK/CAEK cannot be derived.	key-derivation key (after the session completes)	Holds	medium	yes
U02	**Binding of the CEK to its context.** The CEK the recipient accepts matches the sender's pskid, the recipient identity, the KDF context and the key wrap algorithm.	no compromise	Fails (1 attack)	low	no
U03	**One-sided leaks and sender identity.** Checked separately: the CEK stays secret when only the PSK leaks and when only the key-derivation key leaks; and the boundary of sender authentication.	PSK or key-derivation key (after the session completes)	Fails (1 attack)	medium	unclear
X01	**PSK identifier privacy.** An observer of a message cannot tell which PSK identifier the sender used, that is, the PSK identifier does not tell the observer which parties share a PSK.	no compromise	Attack holds (stated in the documents)	low	yes
X02	**CEK delivery under a broken key transport.** An attacker that holds a session's key-derivation key cannot make the recipient accept a content-encryption key that no sender produced, and cannot make it accept a CEK of the attacker's choosing.	the key-derivation key of a session, obtained by breaking the key transport with a large-scale quantum computer, or by reading the recipient's key transport private key; the network	Holds	medium	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, X02.

The questions asked, and the analyst's answers:

Question	Answer	Rests on
Q1: CEK secrecy when the traditional key transport key leaks and the PSK does not	Yes: with the session's key-derivation key known and the PSK secret, the content-encryption key stays unknown, which is proved and is the point of the mechanism.	U01, K01
Q2: PSK ID, recipient, KDF context and key wrap binding	The recipient, the key-encryption algorithm and the content are bound, but the PSK identifier is not: a message can be accepted under another identifier that selects the same PSK, so an implementation must not read anything from the pskid of an accepted message.	U02, A03, K04, D01, A07
Q3: controls for PSK-only and traditional-key-only leaks, and the sender authentication boundary	Each secret alone leaves the content-encryption key secret while the two together disclose it, as the document says; but there is no sender authentication, since any holder of the PSK can put a message to the recipient.	U03, C03, A03

4 Limits of this analysis

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

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

- One recipient and one message flow: several recipients, a message shared by a group of PSK holders, and keyAgreePSK are outside the analysed scope.
- Key transport, KDF, key wrap and authenticated encryption are idealised black boxes: no lengths, no comparison of algorithm strengths, no block-cipher or counter-mode behaviour, no computational assumptions.
- The pskLength and kdkLength inputs of the KDF context have no counterpart in the model, so the domain separation they provide is not checked.
- The pskid re-labelling finding assumes the same PSK is registered under two identifiers; the source leaves that open.
- Sender signing, certificate and recipient-identifier handling, AuthAttributes and the content type are not modelled.
- An attacker that holds the content-encryption key or the key-encryption key of a session reads that content by definition; those cases are not properties and are not asked.
- No claim about availability, timing, random number generation, algorithm strength or resistance to a quantum computer: the model only says that a broken key transport alone does not disclose the key.

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.

K01: Secrecy of CEK, KEK, KDK and PSK

Un-derivability is checked separately for CEK/CAEK, KEK, the key-derivation key and the PSK; the legitimate knowers are the sender and the recipient holding that PSK.

With only the network the attacker learns none of the content-encryption key, the key-encryption key, the key-derivation key or the PSK: the key-derivation key travels only inside the key transport and the key-encryption key is never sent. The key-derivation key becomes known to an attacker that breaks the key transport, which is what the mechanism assumes.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value medium: The secrecy of these four values is the stated purpose of the mechanism (main:581-600, in particular that only the PSK together with the key-derivation key discloses the content).
- Stated in the documents: yes. Where the documents say it: main:581, main:592
- Rests on: The key transport, the KDF and the key wrap are idealised: no algorithm strength, no lengths, no related-key behaviour is modelled.; One recipient and one message flow; group PSKs and keyAgreePSK are outside the scope.; The pskLength and kdkLength inputs of the KDF context are not modelled.
- Open: Whether the KDF and the wrap algorithm an implementation picks keep this apart in practice is a computational question.
- Next step: Nothing further here; the computational strength of the KDF and of the wrap is outside what ProVerif can say.

K02: Confidentiality of the protected content

CEK/CAEK and the content they protect are visible only to the sender and to the recipient holding the same PSK.

Neither the content nor the content-encryption key reaches an attacker on the network: the content is encrypted and authenticated under a key that is itself wrapped under a key derived from the PSK and a fresh key-derivation key. This matches the statement in the document that a ProVerif proof shows the content-encryption key cannot be learned from the exchange.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value medium: main:692-698 states exactly this result for the mechanism, and main:576-579 carries the CMS enveloped-data considerations over.
- Stated in the documents: yes. Where the documents say it: main:576, main:692
- Rests on: Authenticated encryption is modelled as an ideal encryption plus a MAC under the same key.; The content is one value; the AuthAttributes and the content type are outside the model.; One recipient; a message shared by several recipients is outside the scope.
- Next step: Nothing further here.

A03: Agreement on CEK, pskid and algorithm

Every time the recipient accepts a CEK there is a matching sender run, and both sides agree on the identity, the pskid, the keyEncryptionAlgorithm and the CEK.

Against the network, the recipient's acceptance is always matched by a sender run, and the two agree on the key, the algorithm and the content. The pskid, however, is not covered: a message can be accepted under another identifier of the same PSK. Against an attacker that belongs to the group sharing the PSK there is no matching sender at all, because any holder of the PSK can build a message the recipient accepts, and nothing in the flow says who sent it.

- Result: Fails (2 attacks); answers what is asked: partly; usable with caveats
- Value medium: The documents give the general rule that a key management technique must itself provide data origin authentication and that RSA key transport does not (rfc5652:2864-2872), and that a key shared by more than two parties gives no origin authentication (rfc5652:2874-2877); main:649-659 discusses a group PSK only for eavesdropping. No document says that a PSK holder can put content to a recipient as if from the sender.
- Stated in the documents: unclear. Source text it rests on: rfc5652:2864, rfc5652:2874, main:649, s0649, s0652
- Rests on: The pskid attack assumes the recipient has the same PSK registered under a second identifier, which the source leaves open.; pskid is deliberately kept out of the KDF context in the model, as the source does: the context carries the PSK value (s0038-s0045).; One recipient only; the recipient identifier is not a separate field because the key transport binds the message to the recipient's public key.
- Open: Whether the sender's identity is meant to be inferred at all from a keyTransPSK message.
- Next step: A statement in the specification of what the recipient may infer about the origin would settle the reading.

A04: One acceptance per message instance

Every time the recipient accepts a CEK it corresponds to a different sender run, and the same message instance is accepted at most once.

The same message can be delivered twice and accepted twice: the flow has no message identifier, nonce or counter, so a recipient that treats an accepted key as a one-time event is not protected. For plain enveloped content the repeat only re-delivers the same content.

- Result: Fails (1 attack); answers what is asked: yes; usable with caveats
- Value low: I searched all five documents for replay, replay protection, nonce and freshness; the word replay does not occur anywhere and no message carries a value that could reject a repeat.
- Stated in the documents: no
- Rests on: The recipient is replicated, which is what lets the replay be seen; no recipient state on earlier messages is modelled because the protocol has none.; Whether a repeat is harmful is an application question the protocol does not answer.
- Open: Whether the specification intends an accepted key to be usable once.
- Next step: The editors should say whether an application may rely on one acceptance per message.

K04: Domain separation of derived KEKs

KEKs derived under different pskid, keyMgmtAlgType, keyEncryptionAlgorithm or lengths must differ.

The key-encryption key is bound to the PSK, the key management algorithm type and the key-encryption algorithm, so a message cannot be accepted under another algorithm. It is not bound to the pskid: the identifier selects the PSK at the recipient and does not enter the derivation, so two identifiers that select the same PSK give the same key.

- Result: Fails (1 attack); answers what is asked: partly; usable with caveats
- Value low: The source requires that all KDF inputs influence the output (s0048) and lists the context (s0039), but no document observes that the pskid is outside the context and that the separation between contexts therefore rests on the PSK value alone.
- Stated in the documents: no
- Rests on: keyMgmtAlgType is the constant 5 in the model, as the structure fixes it; no keyAgreePSK context (value 10) is modelled, so cross-protocol separation could only be checked in a model that has both.; pskLength and kdkLength are inputs of the context in the source but have no counterpart in a symbolic model.; The pskid attack assumes one PSK registered under two identifiers.
- Open: Separating the two key management types (5 against 10) is beyond this model.; The separation given by the two length inputs is not modelled.
- Next step: Model both recipient-info types in one model, or check the length inputs on a model that carries lengths.

D01: Integrity of encryptedKey and context

The encryptedKey the recipient accepts, together with the pskid, kdfAlgorithm and keyEncryptionAlgorithm, corresponds to a genuine send by the sender, and has not been tampered with.

A modified encryptedKey, content or key-encryption algorithm is rejected: the recipient's acceptance is matched by a sender run with the same values. The pskid field is not covered by that integrity, so it can be changed to another identifier of the same PSK and the message is still accepted.

- Result: Fails (1 attack); answers what is asked: partly; usable with caveats
- Value low: main:576-579 carries the authenticated-enveloped-data considerations over, which cover the encrypted content; the source does not say that pskid is outside the integrity of the derived key.
- Stated in the documents: unclear
- Rests on: The key wrap and the authenticated encryption are ideal: no partial modification of a block cipher mode is modelled.; Only the fields the model carries are checked: pskid, the key-encryption algorithm, the key transport value, the wrapped key, the content and its tag.
- Next step: Nothing further; the pskid part is answered by the binding result.

A07: pskid selects the recipient's own PSK

The pskid the recipient accepts matches the identity of the recipient that actually holds that PSK, so another recipient's credentials are not mistaken for its own.

The recipient uses the PSK from its own store that the received pskid selects, so a message protected with another PSK is not accepted. The identifier itself is not authenticated: a recipient that holds the same PSK under a second identifier accepts the message under that identifier, so a pskid cannot be relied on as a name of the PSK that was used.

- Result: Fails (1 attack); answers what is asked: partly; usable with caveats
- Value low: main:189-196 requires an identifier and says unique identifiers are best, and that a recipient may hold several PSKs under one identifier; it does not say what a recipient may infer from the identifier of an accepted message.
- Stated in the documents: no. Source text it rests on: main:189
- Rests on: One recipient with one store; a group of recipients sharing a PSK is outside the model.; The identifier-to-PSK mapping is modelled as configuration of the recipient, as the source leaves it.

- Next step: A rule on how identifiers are assigned to PSKs would settle it.

C03: Session isolation on KDK or PSK leak

When one session's key-derivation key or PSK leaks, the CEKs of other sessions remain secret.

A key-derivation key that leaks damages only its own session, because every session has its own; a leaked PSK alone discloses no content-encryption key, because the key transport still has to be broken. Only the PSK together with a session's key-derivation key gives that session's key, which is what the document states.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value medium: main:581-600 states that the compromise of the key-encryption key or of the combination of the PSK with the key-derivation key discloses the contents protected with that keying material, and that the PSK alone does not.
- Stated in the documents: yes. Where the documents say it: main:581, main:592
- Rests on: The leak is modelled per session, as the document describes a quantum attacker that discovers one key-derivation key.; Each session is a fresh run of the sender with its own key-derivation key and content-encryption key.
- Next step: Nothing further here.

U01: CEK secret when the KDK leaks

When the key-derivation key leaks and the PSK does not, the CEK/CAEK cannot be derived.

An attacker that has a session's key-derivation key still does not learn that session's content-encryption key, because the key-encryption key also needs the PSK. This is the promise of the mechanism: breaking the key transport, today or with a quantum computer later, is not enough to read the content.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value medium: main:602-605 says a large-scale quantum computer can discover the key-derivation key and main:592-600 says only the PSK together with the key-derivation key discloses the content.
- Stated in the documents: yes. Where the documents say it: main:592, main:602
- Rests on: The KDF is a one-way function of its inputs with no algebraic structure; the PSK enters as a distinct argument.; The key transport is ideal, and the attacker is given the key-derivation key directly, which is the strongest form of a broken key transport.; No lengths and no algorithm strength are modelled.
- Next step: The computational security of the KDF against an attacker that knows the key-derivation key is outside what ProVerif can say.

U02: Binding of the CEK to its context

The CEK that the recipient accepts is consistent with the sender's pskid, the recipient identity, the KDF context and the key wrap algorithm.

The content-encryption key is bound to the recipient, to the key-encryption algorithm and to the content, so a tampered or misdirected message is rejected. It is not bound to the pskid, so a message can be accepted under another identifier that selects the same PSK.

- Result: Fails (1 attack); answers what is asked: partly; usable with caveats
- Value low: The context list in s0039 does not contain the pskid, and no document states what that leaves unbound.

- Stated in the documents: no
- Rests on: The recipient identity is bound only through the key transport to its public key; the RecipientIdentifier field is not modelled separately.; A single recipient, so redirection between recipients is not examined.
- Open: Misdirection between two recipients that share a PSK is not modelled.
- Next step: A model with two recipients that share a PSK would settle the recipient part of the binding.

U03: One-sided leaks and sender identity

Separately check that the CEK stays secret when only the PSK leaks and when only the key-derivation key leaks, and check the sender-authentication boundary.

Either secret alone leaves the content-encryption key secret: the PSK without the key-derivation key, and the key-derivation key without the PSK. What the PSK does not give is any assurance about who sent the content: every holder of the PSK can put a message to the recipient, so the recipient's only conclusion is that some holder of the PSK sent it.

- Result: Fails (1 attack); answers what is asked: yes; usable with caveats
- Value medium: main:649-659 says knowledge of the PSK does not by itself enable eavesdropping and says nothing about who can send; rfc5652:2864-2877 gives the general rule that key transport does not provide data origin authentication and that a key shared by many gives none.
- Stated in the documents: unclear. Source text it rests on: rfc5652:2864, main:649, s0649, s0652
- Rests on: The group PSK is a value the attacker knows, which is how the document describes a PSK shared by a group.; The forged message is built with the public key of the recipient, which the model publishes as a certificate.; Sender signing is outside the model, as the task puts it outside the scope.
- Next step: A sentence in the specification on what a recipient may infer about the origin would settle it.

X01: PSK identifier privacy

An observer of a message cannot tell which PSK identifier the sender used, that is, the PSK identifier does not tell the observer which parties share a PSK.

The PSK identifier travels in the clear, so an observer learns which PSK a message was protected with and can group messages by it. The document states this and argues that the recipient identifier was already visible, so the privacy situation does not get worse.

- Result: Attack holds (stated in the documents); answers what is asked: yes; usable
- Value low: main:702-709 states it in so many words.
- Stated in the documents: yes. Where the documents say it: main:702
- Rests on: The privacy question is asked for the identifier only, on two identifiers that select the same PSK so that both runs complete; no other distinguisher is examined.
- Next step: Nothing further; the document already says it.

X02: CEK delivery under a broken key transport

An attacker that holds a session's key-derivation key cannot make the recipient accept a content-encryption key that no sender produced, and cannot make it accept a CEK of the attacker's choosing.

An attacker that has broken the key transport and holds a session's key-derivation key cannot make the recipient accept a key of its own choosing: without the PSK it cannot produce the key-encryption key, so the delivery of the content-encryption key is intact against it.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value medium: main:692-698 states that a ProVerif proof shows a quantum attacker cannot disrupt the delivery of the content-encryption key.
- Stated in the documents: yes. Where the documents say it: main:692
- Rests on: The attacker is given the key-derivation key directly instead of a key transport break, which is the strongest form.; The key transport is ideal, so the recipient always recovers the key-derivation key it was sent.
- Next step: Nothing further here.

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
K01	proved		k01_cek	checked		4 proved
K02	proved		k02_content	checked		2 proved
A03	proved		a03_agree	checked	a03_pskid (depends_on_ambiguity); a03_forge (holds_in_source)	1 proved, 2 attack
A04	none				a04_once (holds_in_source)	1 attack
K04	proved		k04_alg	checked	a03_pskid (depends_on_ambiguity)	1 proved, 1 attack
D01	proved		a03_agree	checked	a03_pskid (depends_on_ambiguity)	1 proved, 1 attack
A07	proved		a03_agree	checked	a03_pskid (depends_on_ambiguity)	1 proved, 1 attack
C03	proved		u03_psk	checked		2 proved
U01	proved		u01_kdk	checked		1 proved
U02	proved		k04_alg	checked	a03_pskid (depends_on_ambiguity)	2 proved, 1 attack
U03	proved		u03_psk	checked	a03_forge (holds_in_source)	2 proved, 1 attack
X01	none				x01_privacy (holds_in_source)	1 attack
X02	proved		x02_delivery	checked		1 proved

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

target	claim	attacker	answer	query	control
K01	The content-encryption key (CEK/CAEK) of a session cannot be derived by an attacker that has only the network; there is no compromise in this query.	`network`	proved	k01_cek	checked
K01	The key-encryption key (KEK) cannot be derived by an attacker that has only the network; it never leaves the two parties.	`network`	proved	k01_kek	checked
K01	The key-derivation key (KDK) cannot be derived by an attacker that has only the network: it becomes known only where the session's KDK leaks.	`network`	proved	k01_kdk	checked
K01	The PSK cannot be derived by an attacker that has only the network.	`network`	proved	k01_psk	checked
K02	The content protected under the CEK cannot be derived by an attacker that has only the network.	`network`	proved	k02_content	checked
K02	The content-encryption key (CEK/CAEK) that protects the content is visible only to the sender and to the recipient that holds the PSK.	`network`	proved	k01_cek	checked

target	claim	attacker	answer	query	control
A03	Every acceptance of a CEK by the recipient is matched by a sender run, and sender and recipient agree on the CEK, the content and the key-encryption algorithm in the KDF context.	`network`	proved	a03_agree	checked
A03	The pskid the recipient accepts is the pskid the sender used: the recipient does not accept a message labelled with another identifier under which it holds the same PSK.	`network`	attack (depends_on_ambiguity)	a03_pskid	
A03	Every acceptance of a CEK is matched by a sender run, also when the attacker is a member of the group that shares the PSK. Attacker beyond the network: the PSK value itself.	`psk_holder`	attack (holds_in_source)	a03_forge (the control of x02_delivery)	
A04	A recipient accepts a given message instance of one sender run at most once, so every acceptance corresponds to a different sender run.	`network`	attack (holds_in_source)	a04_once	
K04	A KEK derived under a different pskid is a different key: a KEK is never used in a context labelled with another pskid.	`network`	attack (depends_on_ambiguity)	a03_pskid	
K04	A KEK derived under a different keyEncryptionAlgorithm is a different key: the recipient accepts the CEK only with the algorithm the sender put in the KDF context.	`network`	proved	k04_alg	checked
D01	The encryptedKey the recipient accepts, and the key-encryption algorithm it is accepted with, belong to a genuine send of the sender and are not modified on the way.	`network`	proved	a03_agree	checked
D01	The pskid carried with the accepted encryptedKey is the one the sender used and has not been modified.	`network`	attack (depends_on_ambiguity)	a03_pskid	
A07	The recipient decrypts with the PSK its own store holds for the received pskid: an acceptance is always matched by a sender run that used a PSK of that store, so no other party's PSK is used.	`network`	proved	a03_agree	checked
A07	The pskid the recipient accepts names the identifier the sender used, so a recipient does not accept a message under another identifier of its store.	`network`	attack (depends_on_ambiguity)	a03_pskid	
C03	A leaked PSK alone discloses no CEK of any session; a session's CEK is disclosed only if that session's key-derivation key is also known. Attacker beyond the network: the PSK value.	`psk_holder`	proved	u03_psk	checked
C03	A leaked key-derivation key compromises only the session it belongs to: a session's CEK becomes known only if that very session's key-derivation key is leaked, so every other session keeps its CEK. Attacker beyond the network: the key-derivation key of one session.	`kdk_leak`	proved	u01_kdk	checked
U01	The CEK of a session stays unknown to an attacker that holds that session's key-derivation key, as long as the PSK is not known. Attacker beyond the network: the session's key-derivation key.	`kdk_leak`	proved	u01_kdk	checked

target	claim	attacker	answer	query	control
U02	The CEK is bound to the recipient: the key-derivation key is transported to the recipient's own public key, and an acceptance is matched by a sender run under that recipient's PSK.	`network`	proved	a03_agree	checked
U02	The CEK is bound to the pskid the sender used: the recipient never accepts the CEK under a pskid the sender did not send.	`network`	attack (depends_on_ambiguity)	a03_pskid	
U02	The CEK is bound to the KDF context and to the key-encryption algorithm: the recipient accepts it only under the algorithm the sender put into the derivation.	`network`	proved	k04_alg	checked
U03	A leaked PSK alone leaves the CEK secret; only the PSK together with the key-derivation key discloses it. Attacker beyond the network: the PSK value.	`psk_holder`	proved	u03_psk	checked
U03	A leaked key-derivation key alone leaves the CEK secret. Attacker beyond the network: the session's key-derivation key.	`kdk_leak`	proved	u01_kdk	checked
U03	The recipient has no assurance of which sender produced a message: every acceptance of a CEK is matched by a sender run only as long as the attacker is not a member of the group that shares the PSK. Attacker beyond the network: the PSK value.	`psk_holder`	attack (holds_in_source)	a03_forge (the control of x02_delivery)	
X01	An observer cannot tell which PSK identifier a sender used: the pskid does not tell the observer which parties share a PSK.	`network`	attack (holds_in_source)	x01_privacy	
X02	An attacker that holds a session's key-derivation key cannot make the recipient accept a CEK that no sender produced, nor a CEK of its own choosing. Attacker beyond the network: the session's key-derivation key.	`kdk_leak`	proved	x02_delivery	checked

The prover never says that a property of this kind (the attacker cannot tell which of two values is used) is false. It gave no proof and a trace on which the attacker tells the values apart, and the attack is recorded on that trace: x01_privacy.

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-20261114T231408, 2026-11-14. Status: finished. Model: deepseek-v4-1-flash-261017. Steps: 46, prover 8 s, wall 12 min, cost 0.105 USD.

ProVerif: 123 proofs run, 117 decided, 6 without an answer, 4 s, 30 answers recorded.

What was analysed

Protocol	CMS key transport plus PSK
Version	rfc8696
Analysed	A single-recipient, single-message flow fixing RFC8696 KeyTransPSKRecipientInfo; the KEK combines a high-entropy independently pre-shared PSK of the peer with a random secret of traditional key transport. It uses the specified KDF/context and AuthEnvelopedData; PSK transport, group sharing and sender signatures are out of scope.
Not analysed	The symbolic model does not prove computational or post-quantum security of the algorithms; attack traces hold only under explicit assumptions and are not directly equivalent to a specification flaw. Every property must have a reachability/positive control.
Source text	48 numbered passages (10358 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	3	keyTransPSK
main	4	keyAgreePSK
main	5	Key Derivation

The purpose of the protocol as the source step located it in the documents: A single-recipient, single-message flow fixing RFC8696 KeyTransPSKRecipientInfo; the KEK combines a high-entropy independently pre-shared PSK of the peer with a random secret of traditional key transport. It uses the specified KDF/context and AuthEnvelopedData; PSK transport, group sharing and sender signatures are out of scope.

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
K01	Secrecy of CEK, KEK, KDK and PSK	The content-encryption key (CEK/CAEK), the key-encryption key (KEK), the key-derivation key (KDK) and the PSK are known only to the sender and the recipient that holds that PSK; no one else learns them from the exchange.	The KEK is derived from the PSK and the KDK and must be protected (main:581-590); the KDK travels only inside KeyTransRecipientInfo's encryptedKey (s0015) and the CEK only inside encryptedKey under the KEK (s0016).	main:581, main:592, s0002, s0013, s0015, s0016, s0035, s0038, s0047
K02	Confidentiality of the protected content	The content carried in AuthEnvelopedData under the CEK is readable only by the sender and the recipients that hold the PSK; an attacker on the network learns neither the CEK nor the content.	The CEK is random per content and protects the content; the CMS enveloped-data and authenticated-enveloped-data security considerations continue to apply (main:576-579, s0016).	main:576, main:692, s0002, s0016
A03	Agreement on CEK, pskid and algorithm	Each acceptance of a CEK by the recipient is matched by a sender run, and sender and recipient agree on the PSK identifier, the key-encryption algorithm and the CEK.	The document states no explicit authentication goal; rfc5652:2864-2872 says a key management technique must itself provide data origin authentication and that RSA key transport does not, and rfc5652:2874-2877 says parties sharing a key do not get data origin authentication.	main:576, rfc5652:2864, rfc5652:2874, s0002, s0015, s0016, s0038
A04	One acceptance per message instance	A recipient accepts a given message instance of a sender run at most once; every acceptance corresponds to a different sender run.	No document states replay protection: the word does not occur in any of the five documents. The target asks for the property the single-message flow is expected to have, and the documents are silent on it.	s0002, s0015
K04	Domain separation of derived KEKs	A KEK derived under a different pskid, keyMgmtAlgType, keyEncryptionAlgorithm or length is a different key, so a KEK of one context is never used in another.	The KDF inputs are psk, keyMgmtAlgType (5 for keyTrans), keyEncryptionAlgorithm, pskLength and kdkLength, and all inputs MUST influence the output (s0038-s0048).	main:627, s0013, s0038, s0039, s0042, s0044, s0045, s0048
D01	Integrity of encryptedKey and context	The encryptedKey the recipient accepts, and the pskid, kdfAlgorithm and keyEncryptionAlgorithm fields it is accepted with, belong to a genuine send and have not been modified.	encryptedKey is the CEK encrypted under the KEK (s0016) and the context fields enter the derivation (s0013, s0038); the authenticated-enveloped-data considerations apply (main:576-579).	main:576, s0002, s0013, s0016
A07	pskid selects the recipient's own PSK	The recipient decrypts with the PSK its own store holds for the received pskid; it never accepts a message under another party's PSK and a pskid it accepts names the PSK that was really used.	pskid is the identifier of the PSK used by the sender (s0012); the identifier MUST be assigned and SHOULD be unique, and a recipient may hold several PSKs under one identifier and try each in turn (main:189-196).	main:189, s0012, s0015
C03	Session isolation on KDK or PSK leak	A leaked key-derivation key compromises only its own session; a leaked PSK alone discloses no CEK of any session; earlier and other sessions keep their CEK.	Compromise of the KEK or the key management private key may disclose the contents protected with that keying material, while the PSK alone does not (main:581-600).	main:581, main:592, s0013, s0015, s0016, s0035, s0038
U01	CEK secret when the KDK leaks	The CEK/CAEK of a session stays unknown to an attacker that holds that session's key-derivation key, as long as the PSK is not known; the PSK is what carries the confidentiality.	A large-scale quantum computer can discover the key-derivation key (main:602-605) and the PSK is what protects the content; only the PSK together with the KDK discloses the content (main:592-600).	main:592, main:602, s0013, s0015, s0016, s0035, s0038

ID	Name	About	What the documents say	Where
U02	Binding of the CEK to its context	The CEK the recipient accepts is the one the sender produced for that pskid, that recipient, that KDF context and that key-encryption algorithm.	pskid identifies the PSK (s0012), the KDF context is the DER encoding of CMSORIforPSKOtherInfo (s0038-s0045) and the algorithm is carried in the message (s0013); the documents state no explicit binding goal beyond this construction.	s0012, s0013, s0015, s0016, s0038, s0039
U03	One-sided leaks and sender identity	A leaked PSK alone, or a leaked key-derivation key alone, leaves the CEK secret; and the recipient has no assurance of which sender produced a message, since any holder of the PSK can produce one.	The PSK may be known to a group and knowledge of it does not by itself enable eavesdropping (main:649-659); rfc5652:2864-2877 says key transport with RSA does not provide data origin authentication.	main:649, rfc5652:2864, s0012, s0013, s0015, s0016, s0035, s0038
X01	PSK identifier privacy	The pskid field of KeyTransPSKRecipientInfo and the privacy of the parties that share a PSK, against an observer on the network.	main:702 says an observer can see which parties use each PSK simply by watching the PSK key identifiers; the document states that this privacy is not provided.	main:702
X02	CEK delivery under a broken key transport	The delivery of the CEK to the recipient, against an attacker who has broken the key transport (a large-scale quantum computer, or the recipient's transport private key).	main:692-698 says a ProVerif proof shows that an attacker with a large-scale quantum computer cannot disrupt the delivery of the content-encryption key to the recipient and cannot learn it.	main:692

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

The security statements of the documents, paragraph by paragraph

The host listed the 34 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: 12. Not taken: 22. Which paragraphs are listed follows from the layout of the standard; where each went is the analyst's judgement.

Paragraphs	Where	Taken up as	Why not
P01	main:576-579	K02, D01	
P02	main:581-590	K01, C03, `psk_holder`, `kdk_leak`	
P03	main:592-600	K01, U01, U03, C03, `psk_holder`, `kdk_leak`	
P04	main:602-613	U01, K01, `kdk_leak`	
P05	main:615-625	The prover cannot be asked	It bounds the effective strength by the weaker of two algorithms; a symbolic prover cannot compare algorithm strengths.
P06	main:627-632	The prover cannot be asked	It bounds the strength of the KEK by the strength of the chosen KDF.
P07	main:634-640	Outside the scope	It is about protecting the same content with both a traditional and a PSK-mixing technique; the task fixes the single-recipient keyTransPSK flow and does not model a second key management technique.
P08	main:642-647	Outside the scope	It is about correlating two separate messages that carry the same content; the model has one message flow per run and no cross-message correlation.
P09	main:649-659	U03, C03, A03, `psk_holder`	

Paragraphs	Where	Taken up as	Why not
P10	main:661-670	K01, K02	
P11	main:672-682	The prover cannot be asked	It is about the quality of random number generators, which a symbolic prover assumes perfect.
P12	main:684-690	The prover cannot be asked	It is about algorithms weakening over time, outside what a symbolic prover can say.
P13	main:692-698	K01, K02, A03, X02, `kdk_leak`	
P14	main:702-709	X01, `network`	
P15	rfc5083:308-310	No goal stated	It says the CMS security considerations apply; the goals are in the CMS paragraphs P22-P31.
P16	rfc5083:312-317	The prover cannot be asked	It compares the privacy guarantees of counter mode and CBC mode, a property of the block cipher mode.
P17	rfc5083:325-328	The prover cannot be asked	It is advice on using counter mode with a fresh key and a per-encryption nonce, a property of the authenticated encryption algorithm.
P18	rfc5083:330-337	K02	
P19	rfc5083:339-348	The prover cannot be asked	It is about the quality of random number generators.
P20	rfc5083:350-355	Outside the scope	It is about the message-digest attribute of AuthAttributes enabling content tracking; AuthAttributes and signed attributes are outside the modelled flow.
P21	rfc5083:357-369	The prover cannot be asked	It is about handling unwanted message traffic, a deployment and filtering policy, not a property of the exchange.
P22	rfc5652:2837-2839	No goal stated	It states only that CMS offers signing, digesting, encryption and authentication; no requirement on this flow.
P23	rfc5652:2841-2842	Outside the scope	It is about the signer's private key and masquerade through signatures, and sender signing is outside the task.
P24	rfc5652:2844-2849	C03, K01, U01, `kdk_leak`	
P25	rfc5652:2851-2855	Outside the scope	It is about the message-authentication key of the authenticated-data content type, which is outside the modelled flow.
P26 to P27	rfc5652:2864-2877	A03, `psk_holder`	
P28	rfc5652:2879-2889	The prover cannot be asked	It is about the quality of random number generators.
P29	rfc5652:2891-2903	The prover cannot be asked	It compares the strength of key-encryption and content-encryption algorithms.
P30	rfc5652:2905-2911	The prover cannot be asked	It says algorithms weaken with time; a symbolic prover cannot express that.
P31	rfc5652:2917-2927	Outside the scope	It is about countersignatures and the countersignature attribute, outside the modelled flow.
P32	rfc5869:355-360	No goal stated	It refers to the HKDF paper for the analysis; no requirement on this protocol.
P33	rfc5869:362-371	No goal stated	It refers to the cryptographic analysis of HKDF and its dependence on the hash function; a property of the KDF primitive.
P34	rfc8619:195-199	No goal stated	It refers to the HKDF design analysis; no requirement on this protocol.

The parties and the protected values, one by one

The host listed the 16 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: 6. Without: 10. 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	sender	party	`psk_holder`	
S02	receiver	party	`psk_holder`, `kdk_leak`	
S03	CEK/CAEK	value	Everything lost by definition	A party that holds the content-authenticated-encryption key of a session reads that content by definition; no target asks a property about a session whose CEK the attacker already holds.
S04	KEK	value	Everything lost by definition	The KEK exists to protect the CEK of that session; with it the CEK is lost by definition (main:584-590).
S05	key-derivation key	value	`kdk_leak`	
S06	PSK	value	`psk_holder`	
S07	protected content	value	Everything lost by definition	The protected content itself is the value the confidentiality targets keep; holding it answers them by definition.
S08	pskid	value	Nothing gained	pskid travels in the clear in every message (main:702-709), so an attacker on the network has it already.
S09	keyEncryptionAlgorithm	value	Nothing gained	keyEncryptionAlgorithm is a field of the message and is public.
S10	encryptedKey	value	Nothing gained	encryptedKey is a field of the message and is public.
S11	KDF context	value	`psk_holder`	
S12	context fields	value	Nothing gained	The context fields carried in the message (pskid, keyEncryptionAlgorithm, algorithm identifiers, lengths) are public.
S13	recipient identity	value	Nothing gained	The recipient is identified in the clear by the RecipientIdentifier and by pskid (main:702-709).
S14	CEK	value	Everything lost by definition	The CEK is the value the confidentiality targets protect; a party that holds it reads the content by definition.
S15	key wrap algorithm	value	Nothing gained	keyEncryptionAlgorithm is carried in the message and is public.
S16	sender identity	value	`psk_holder`	

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		

Attacker	Has	Basis	Where
psk_holder	the PSK value itself (a member of the group the PSK is shared with, or a leak of the PSK); the network	main:649-656 says the PSK may be known to a group and that knowledge of it does not inherently enable eavesdropping; main:592-600 says the PSK must be protected and that the PSK alone is not enough, PSK with the transport private key or the KDK is	main:649, main:592
kdk_leak	the key-derivation key of a session, obtained by breaking the key transport with a large-scale quantum computer, or by reading the recipient's key transport private key; the network	main:602-605 (a large-scale quantum computer can discover the key-derivation key) and main:592-600 (compromise of the key transport private key)	main:602, main:592

ID	network	psk_holder	kdk_leak
K01	Holds		
K02	Holds		
A03	Fails	Fails	
A04	Fails		
K04	Fails		
D01	Fails		
A07	Fails		
C03		Holds	Holds
U01			Holds
U02	Fails		
U03		Fails	Holds
X01	Fails		
X02			Holds

Source and targets

Source: 48 units, 10358 bytes, frozen in source-delivery-01.json. Targets: 11 approved (K01, K02, A03, A04, K04, D01, A07, C03, U01, U02, U03); 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-6d92b1ad9595b57db364c5cf5fb1

This step as it reports itself: 1 requests (1 failed), 0.000 USD, 17857 tokens in and 4318 out, 22 s.

Notes on the prover and the model

What the analyst named for targets:

step	targets	features	reason
32	K01, K02, A03, A04, K04, D01, A07, C03, U01, U02, U03, X01, X02	plain_primitives, many_sessions	All targets are answered on a model that uses only idealised symmetric primitives (the KEK derivation as a one-way function with no algebraic laws, authenticated encryption as an ideal encryption plus MAC) and idealised public-key key transport (aenc/adec as a perfect encryption), and that quantifies over any number of sender and recipient sessions. No exclusive-or, Diffie-Hellman or other algebra is used, and no value is kept and changed across steps, so no target rests on a changed value; the cryptographic strength of the key transport, the KDF and the algorithms is not modelled at all.

The analyst's closing note

RFC 8696 keyTransPSK, single recipient and single message, was modelled in ProVerif with ideal key transport, KDF, key wrap and authenticated encryption, unbounded sessions, and the KDF context built as the source lists it: the PSK value, keyMgmtAlgType = 5 and the key-encryption algorithm, deliberately without pskid.

What is established. Ten of the thirteen properties are proved, each with a control that fails. The content-encryption key, the key-encryption key, the key-derivation key and the PSK stay out of reach of the network attacker, and so does the content. The central promise holds: an attacker that has the session's key-derivation key, which is what breaking the key transport or a quantum computer gives, still does not learn the content-encryption key, because the key-encryption key also needs the PSK; a leaked PSK alone gives nothing either; only the two together disclose a session's key. One session's key-derivation key damages only that session. The recipient is bound to the key, the key-encryption algorithm and the content, and changed values are rejected. The document's own claim that a quantum attacker cannot disrupt the delivery of the content-encryption key is reproduced.

What fails. The PSK identifier is not an input of the KDF, so it is not covered by the derived key: where the recipient holds the same PSK under a second identifier, an attacker relabels pskid and the message is accepted under the wrong identifier. Any holder of the PSK can build a complete message for the recipient, because the flow names no sender, so content authenticity rests on the PSK alone. A message carries nothing fresh, so the same message can be accepted twice. The privacy of the pskid fails exactly as section 8 of the document says it does.

What blocks the rest. The proofs rest on idealised primitives, on one recipient, and on lengths being outside the model; the key-management-type separation (5 against 10) and the two length inputs of the context could not be asked. The pskid fix is checked on a second model - adding pskid to CMSORIforPSKOtherInfo removes that attack, keeps the normal run and every other recorded answer - while the replay fix needs recipient state the single-message model cannot hold, and the sender-authentication gap can only be closed by pairwise PSKs or signatures, outside the analysed scope. No value is kept and changed across steps, so no bounded model and no state review applied; Tamarin+ Agent would be needed for the computational questions and for the multi-recipient and multi-mechanism cases.