Security analysis of CMP KEM-based MAC protection

This report in another language: 中文.

rfc9810 · ProVerif+ Agent report

Documents	Internet X.509 Public Key Infrastructure -- Certificate Management Protocol (CMP) (protocol definition)
Properties examined	13: 9 hold, 4 fail (6 attacks)
Findings	1

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

CMP can protect a PKI management message with a MAC keyed from a KEM shared secret, so that a PKI entity that holds a KEM key pair can authenticate the messages it sends without a signature. Alice holds the KEM key pair; Bob holds her authentic public KEM key, for instance from a validated KEM certificate. The exchange modelled here is one PKI management operation, not the certificate issuing exchange that CMP otherwise carries.

Bob encapsulates with Alice's public key and sends the ciphertext with a transactionID, his senderNonce and the optional kemContext. Alice decapsulates, derives the MAC key from the shared secret with the transactionID and the kemContext as context, and MACs the message. Bob derives the same key and verifies the MAC, and by that verification he authenticates Alice for this operation.

The parties:

- **Alice**: holds the KEM key pair and protects the messages it sends. Holds: sk (KEM private key), pk.
- **Bob**: holds Alice's authentic public KEM key, encapsulates and verifies. Holds: pk (Alice's KEM key, from a validated certificate).

Figure 1 shows the exchange as the documents specify it.

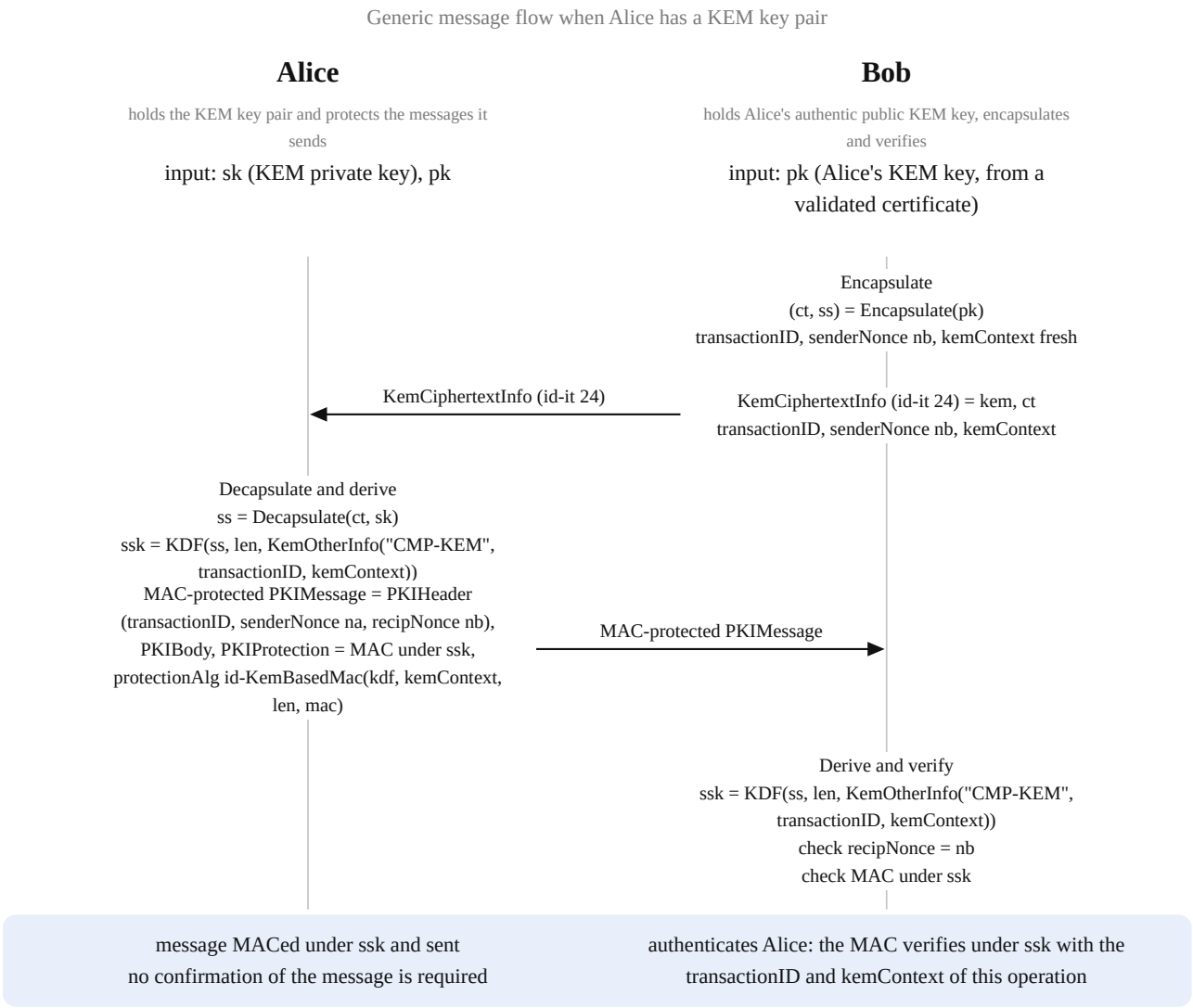


Figure 1. The protocol as specified.

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

The KEM-based MAC protection gives what the documents ask of it against an attacker on the network: Bob's acceptance is tied to a real Alice run, to his own transactionID, nonces and kemContext, and to the message fields, and the shared secret and the MAC key stay secret for any number of operations. One gap is new: inside a single operation the nonce check does not tell the messages apart, so a protected message recorded from the network and replayed as the next message of the same operation is accepted a second time. The documents say the nonces protect against replay and require a fresh encapsulation per operation, but they never require the recipient to record the senderNonces it has accepted, and they reuse one MAC key for the whole operation. A record of accepted senderNonces at the recipient removes the attack; the change was checked on the unrolled model and leaves every

other result as it is. The forward secrecy result and the effect of a leaked KEM private key that the analysis reproduces are stated in the documents themselves. All results hold for an ideal symbolic KEM, KDF and MAC, and for one sender-recipient pair in one direction.

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 protected message replayed inside the same operation is accepted twice

Kind	Weakness of the protocol as specified
New	Unclear: the documents state it in part, or could not be searched
Value	medium
Properties	A04, R02 (section 3)
Source text	s0014, s0075
Shown by	ProVerif attack queries `a04_b`, `r02_b`, `a04_inj`; Figures 2 and 3

What happens. Bob derives one MAC key per management operation and uses it for every protected message Alice sends in that operation, and each of those messages carries the same recipNonce, Bob's senderNonce. The recipient compares only that nonce, which cannot change while Bob sends no further message, and the MAC verifies under the reused key. A message recorded from the network therefore passes both checks when it is sent again as the next message of the operation, and the action its PKIBody requests is carried out a second time.

What the attacker needs. Only the network: the attacker records one protected message of an operation that carries more than one message from Alice, withholds a later one and replays the recorded one. It needs no key and no broken primitive.

What the attacker gains. A message meant to be processed once is processed twice, and Bob cannot distinguish the replay from a genuine later message of the operation. The loss is the replay protection that Section 5.1.1 attributes to the nonces; the KEM secrets and the MAC key are not affected.

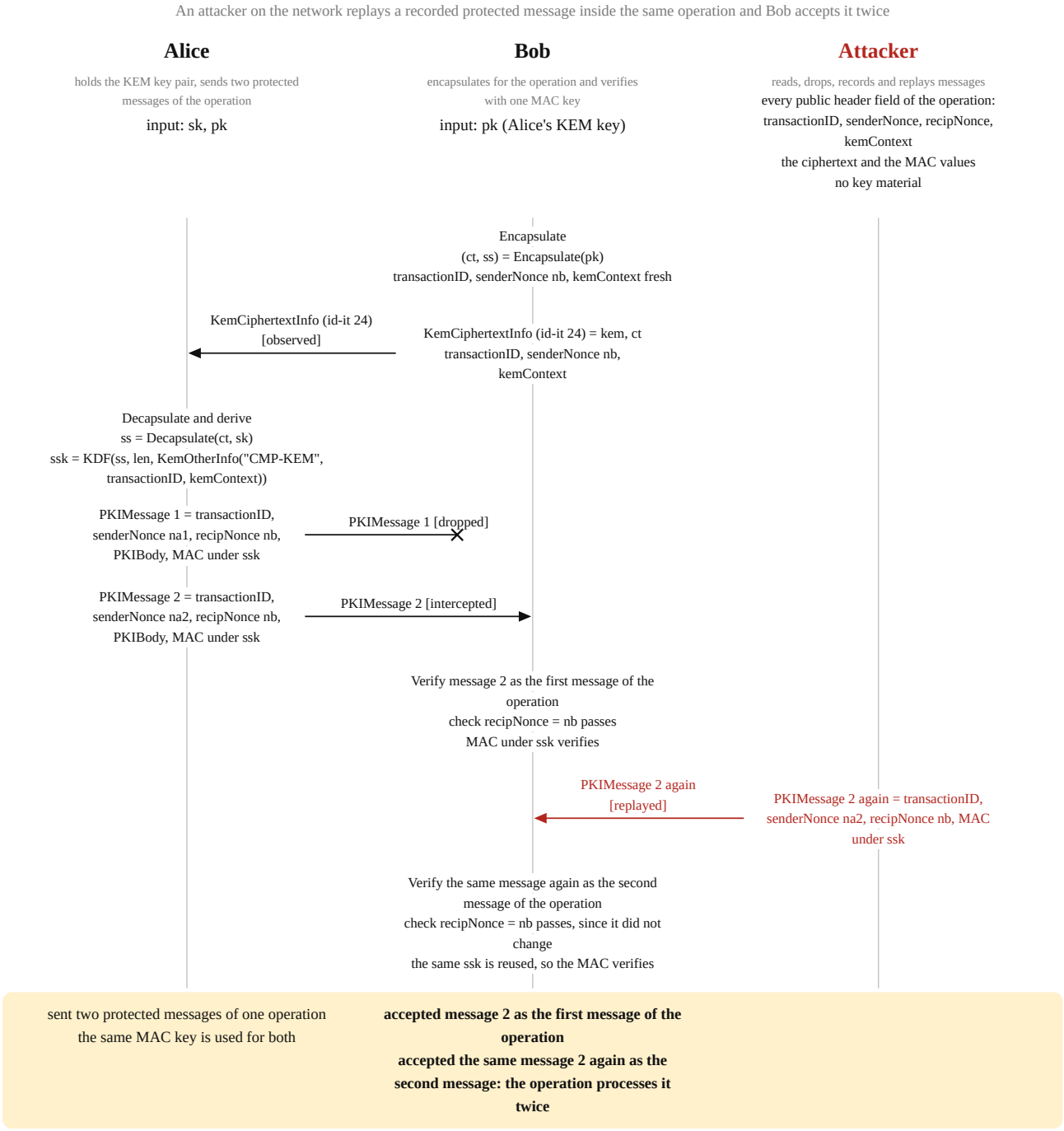


Figure 2. An attacker on the network replays a recorded protected message inside the same operation and Bob accepts it twice. Property A04; attack queries a04_b, a04_inj.

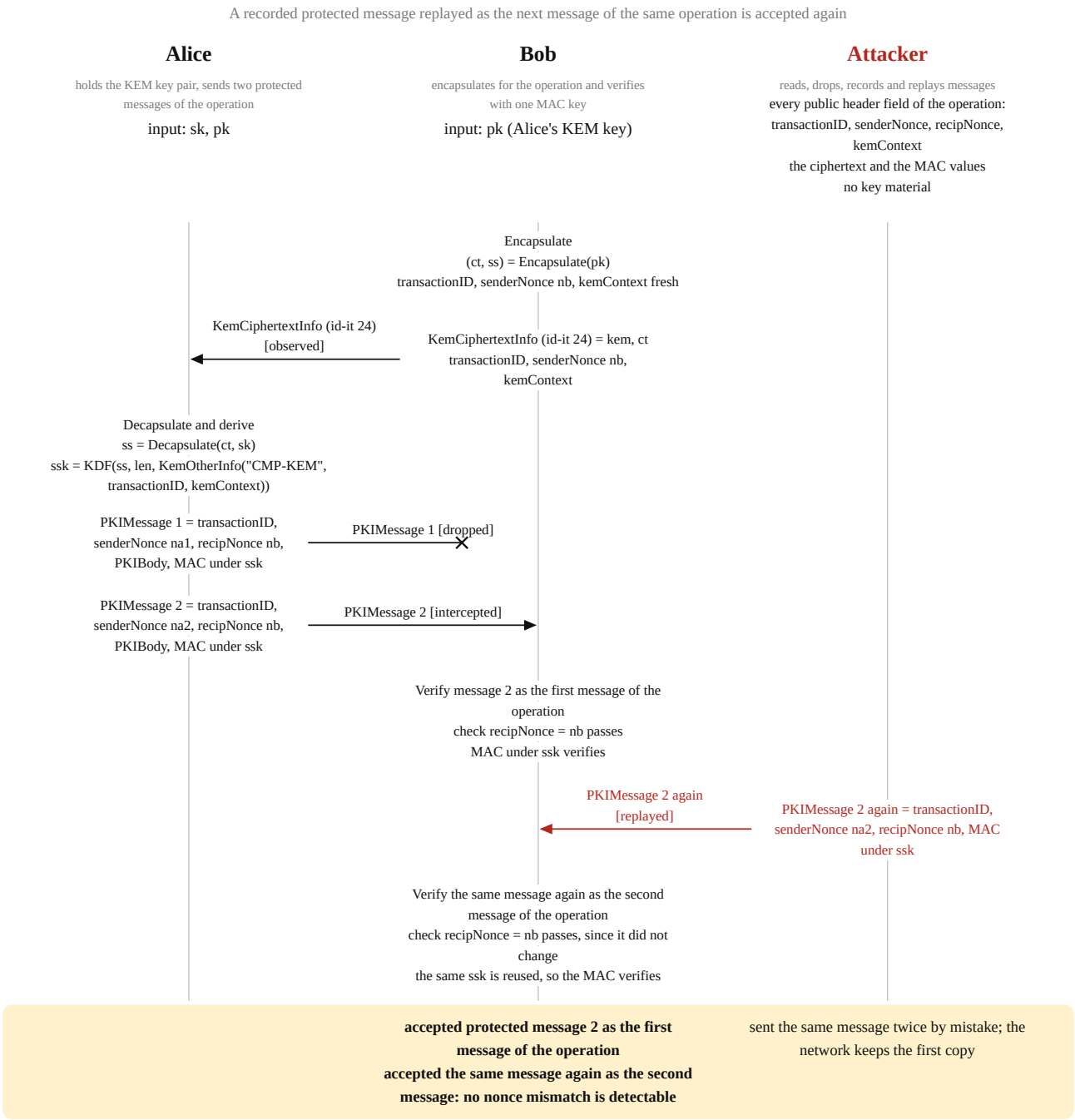


Figure 3. A recorded protected message replayed as the next message of the same operation is accepted again. Property R02; attack query r02_b.

Proposed change. Require the recipient of a MAC-protected PKIMessage to keep the senderNonce values accepted in the transaction and to reject a message whose senderNonce was already accepted, with failInfo badSenderNonce. The change was made on the unrolled two-message model: both attack queries are verified there and no other recorded answer changed.

Check of the change with ProVerif. The change was put into a copy of the model: The recipient of a MAC-protected PKIMessage keeps the senderNonce of the first protected message it accepts in a transaction and MUST reject a later message of the same transaction whose senderNonce repeats a value already accepted (failInfo badSenderNonce). On the changed model ProVerif proves `a04_b`: the attack is no longer possible there. Of the other

recorded answers, 0 still hold and 1 changed: `r02_b` (R02, was attack) is now verified. 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/0028-fix-nonce.pv`.

Check of the change with ProVerif. The change was put into a copy of the model: The recipient of a MAC-protected PKIMessage keeps the senderNonce of the first protected message it accepts in a transaction and MUST reject a later message of the same transaction whose senderNonce repeats a value already accepted (failInfo badSenderNonce). On the changed model ProVerif proves `r02_b`: the attack is no longer possible there. Of the other recorded answers, 0 still hold and 1 changed: `a04_b` (A04, was attack) is now verified. 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/0028-fix-nonce.pv`.

No figure was given for these attacks of a finding: A04: r02_b, R02: a04_b, R02: a04_inj.

3 Properties examined

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

ID	Property	Compromise assumed	Result	Value	In the documents
A03	**Bob authenticates Alice.** Every time Bob successfully verifies a MAC-protected message from Alice, there exists a real run of Alice in which both parties agree on the roles, the identities, and the data of this management operation (transactionID, nonce, kemContext, etc.).	no compromise	Holds	medium	yes
A04	**distinct Alice runs per acceptance.** Every successful MAC verification by Bob corresponds to a distinct run of Alice; the same protected message cannot be accepted by Bob twice.	no compromise	Fails (2 attacks)	medium	unclear
D01	**MAC binds the message fields.** For every MAC-protected PKIMessage that Bob accepts, its PKIHeader, PKIBody, transactionID, nonce and kemContext all correspond to a real send by Alice and have not been tampered with by the attacker.	no compromise	Holds	medium	yes
K01	**KEM secrets stay secret.** The attacker cannot derive ss, ssk or Alice's KEM private key sk from the public ct, the protected messages and the public keys.	no compromise	Attack holds (stated in the documents)	medium	yes
K04	**key separation between operations.** The ssk derived for different PKI management operations and different directions are mutually distinct; reusing ssk within the same operation is allowed.	no compromise	Holds	medium	yes
R02	**no replay within an operation.** Within a single management operation, Bob accepts the same logical message at most once; a replayed old message is rejected because the nonce does not match.	no compromise	Fails (1 attack)	medium	unclear
R01	**fresh KEM evidence and nonces.** The nonces and KEM encapsulation evidence that Bob accepts are newly generated for this management operation; old evidence cannot be taken as valid authentication for this one.	no compromise	Holds	medium	yes

ID	Property	Compromise assumed	Result	Value	In the documents
A06	**key confirmation to Bob.** When Bob's MAC verification succeeds, he confirms that Alice has derived and holds the same ssk; this confirmation is bound to this session and context.	no compromise	Holds	medium	yes
A08	**operation context consistency.** The transactionID, kemContext and algorithm parameters carried by the protected message that Bob accepts are consistent with the encapsulation context that Bob actually sent in this management operation.	no compromise	Holds	medium	yes
A03.10	**Bob authenticates the KEM key holder.** Every time Bob successfully verifies a MAC, there exists a real run of Alice holding Alice's KEM private key, and both parties agree on the identities and the session data.	no compromise	Holds	medium	yes
D01.11	**header and body binding.** In the protected message that Bob accepts, the PKIHeader, PKIBody, transactionID and nonce all correspond to a real send by Alice and have not been replaced.	no compromise	Holds	medium	yes
U01	**no ss/ct reuse across operations.** ss and ct are not reused between different management operations; Bob rejects messages that belong to another transaction or that he has already accepted.	no compromise	Holds	medium	yes
X01	**forward secrecy of the MAC key.** An attacker who obtains Alice's KEM private key only after a PKI management operation has finished cannot derive the ssk of that operation, i.e. the recorded ciphertext and protected message do not give the past key away.	Alice's KEM private key sk, obtained after the operation has completed	Attack holds (stated in the documents)	low	yes

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

Goals the documents state that the drafted list of properties did not have, added by the analyst (Appendix D): X01.

The questions asked, and the analyst's answers:

Question	Answer	Rests on
Q1: Bob's authentication of the holder of Alice's KEM private key	Yes: against an attacker on the network, Bob's successful verification implies a run of the party holding Alice's KEM private key with the same transactionID, nonces and kemContext; the only exception is that the same genuine message can be counted twice inside one operation.	A03, A03.10, A04, R02
Q2: MAC binding to PKIHeader/PKIBody, transactionID and nonce	The MAC binds them: Bob accepts only messages whose transactionID, recipNonce and kemContext are his own and whose body and senderNonce are those Alice sent, so a modified field is refused.	D01, D01.11, A08
Q3: no ss/ct reuse across management operations, cross-transaction replay check	Across operations neither ss nor ct is reused and a message carrying another transactionID is refused; the gap is a replay inside one operation, where the same message is accepted a second time.	U01, K04, R02, A04

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:

- Only the generic KEM-based MAC message-protection flow is modelled. Certificate issuance, proof of possession, RA registration and the other CMP message types are out of scope.
- The KEM, the KDF and the MAC are ideal symbolic primitives. Nothing is proved about IND-CCA2 security, the strength of the KDF or the unforgeability of the MAC.
- kemContext is modelled as a value Bob chooses and sends with the ciphertext; the source leaves its content to the algorithm and does not say who fixes it, so a deployment that takes it from the peer is not covered.
- Bob is given Alice's authentic public KEM key and only that one pair of parties exists in the model, so no confusion between two senders or two KEM certificates is examined.
- Only protected messages from Alice to Bob in one direction are modelled; the flow also allows the responder to apply the procedure with its own KEM key pair in the other direction.
- Replay inside an operation is analysed on an operation unrolled to two protected messages and with an opaque PKIBody; the real effect of a repeated message depends on the CMP message type.
- Availability, timing, message size and the quality of the random number generator are not modelled.

What the host records:

- The answers of A04, R02 rest on an operation the model simplifies (Appendix D).

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

Appendix A: Reading of each property

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

A03: Bob authenticates Alice

Every time Bob successfully verifies a MAC-protected message from Alice, there exists a real run of Alice in which both parties agree on the roles, the identities, and the data of this management operation (transactionID, nonce, kemContext, etc.).

Against an attacker on the network Bob accepts a protected message only if Alice ran with the same transactionID, senderNonce, recipNonce, kemContext and body. The MAC key cannot be produced without Alice's KEM private key, so acceptance names Alice and this operation.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value medium: The flow states that Bob authenticates Alice by verifying the MAC.
- Stated in the documents: yes. Where the documents say it: s0074
- Rests on: the KEM, the KDF and the MAC are ideal symbolic primitives; one sender-recipient pair: the peer is fixed by whose public KEM key Bob holds; one direction only
- Next step: Model two senders with different KEM certificates to test what Bob's acceptance says about which identity it names.

A03.10: Bob authenticates the KEM key holder

Every time Bob successfully verifies a MAC, there exists a real run of Alice holding Alice's KEM private key, and both parties agree on the identities and the session data.

Bob's successful verification implies a run of the party that holds Alice's KEM private key, with the same transactionID, nonces and kemContext. An attacker on the network cannot produce such a message without that key.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value medium: The documents define Alice as the entity possessing the KEM key pair who provides the MAC-based protection.
- Stated in the documents: yes. Where the documents say it: s0047, s0074
- Rests on: ideal KEM and MAC; the identity of the peer is given by which public KEM key Bob holds
- Next step: As for the authentication target, a model with two key holders would settle identity confusion.

A04: distinct Alice runs per acceptance

Every successful MAC verification by Bob corresponds to a distinct run of Alice; the same protected message cannot be accepted by Bob twice.

Across operations each acceptance corresponds to a distinct Alice run, because the senderNonce and the shared secret are fresh for every operation. Inside one operation the same protected message is accepted twice, as the unrolled model shows: the recipNonce stays the same and the MAC key is reused.

- Result: Fails (2 attacks); answers what is asked: yes; usable with caveats

- Value medium: Section 5.1.1 says the nonces protect against replay and Section 8.8 requires a fresh encapsulation per operation, but the documents do not state that a message repeated inside one operation is accepted.
- Stated in the documents: unclear. Source text it rests on: s0014, s0097
- Rests on: replay inside an operation is analysed on a two-message unrolling with an opaque PKIBody; the model gives Bob a fresh transactionID and a fresh encapsulation per operation
- Open: the harm of the repeated message depends on the PKIBody action, which is not modelled
- Next step: Model the actual CMP message types of an operation so that the repeated action can be named.

D01: MAC binds the message fields

For every MAC-protected PKIMessage that Bob accepts, its PKIHeader, PKIBody, transactionID, nonce and kemContext all correspond to a real send by Alice and have not been tampered with by the attacker.

Every field of an accepted message is tied to Alice's send by the MAC, and Bob checks transactionID, recipNonce and kemContext against his own operation. A field changed on the way, or a message moved to another operation, does not verify.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value medium: The documents state that protected header information is protected together with the message, and that PKIProtection carries the MAC over the message.
- Stated in the documents: yes. Where the documents say it: s0002, s0048, s0074
- Rests on: ideal MAC: a changed field changes the MAC value; the PKIBody is modelled as a single opaque value, so the per-field structure inside the body is not examined
- Next step: None for the modelled fields.

D01.11: header and body binding

In the protected messages Bob accepts, the PKIHeader, PKIBody, transactionID and nonce all correspond to what Alice actually sent; none of them is replaced.

An accepted message carries Alice's PKIBody and senderNonce and Bob's transactionID, recipNonce and kemContext, all covered by the MAC. A modified header field or body is refused.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value medium: The documents state that the protection covers the message and that the transactionID enters the key derivation.
- Stated in the documents: yes. Where the documents say it: s0002, s0082
- Rests on: ideal MAC; the opaque body is a single value, not a structure of fields
- Next step: None for the modelled fields.

A06: key confirmation to Bob

When MAC verification succeeds, Bob confirms that Alice has derived and holds the same ssk; the confirmation applies to this session and context.

Bob's acceptance confirms that Alice derived the same MAC key for the same operation and context, because the MAC verifies only under that key. The confirmation is only as strong as the secrecy of the key.

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

- Value medium: The documents say Bob verifies the MAC with the shared secret key and in this way authenticates Alice.
- Stated in the documents: yes. Where the documents say it: s0074, s0075
- Rests on: there is no separate key-confirmation message; the confirmation is the successful MAC verification; ideal MAC
- Next step: None.

A08: operation context consistency

The transactionID, kemContext and algorithm parameters carried by the protected messages Bob accepts match the encapsulation context Bob actually sent in this management operation.

Bob accepts only messages that carry his own transactionID and kemContext, and the transactionID enters the key derivation, so a message from another operation or with another context does not verify.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value medium: The documents require the transactionID to be the value of the message carrying the ciphertext, and use it for domain separation of the derived key.
- Stated in the documents: yes. Where the documents say it: s0080, s0082, s0083
- Rests on: kemContext is modelled as a value Bob chooses and sends with the ciphertext; the source leaves its content open; ideal KDF and MAC
- Next step: A deployment where kemContext comes from the peer should be modelled separately.

K01: KEM secrets stay secret

An attacker cannot derive ss, ssk or Alice's KEM private key sk from the public ct, the protected message and the public key.

An attacker on the network cannot derive the shared secret, the MAC key or Alice's KEM private key from the ciphertext, the protected message and the public key, no matter how many operations are run. An attacker who later obtains Alice's KEM private key recovers both the shared secret and the MAC key of a recorded operation; the documents state this as a consequence of such a compromise.

- Result: Attack holds (stated in the documents); answers what is asked: yes; usable with caveats
- Value medium: The documents state that compromise of the KEM private key may disclose all contents protected with that key.
- Stated in the documents: yes. Where the documents say it: rfc9629:568
- Rests on: the KEM is ideal: the ciphertext reveals nothing about the shared secret without the private key; the MAC protects integrity only, so the PKIBody is readable on the wire; the key compromise happens after the operation, in a later phase
- Next step: None.

K04: key separation between operations

The ssk derived for different PKI management operations and different directions are distinct; reusing an ssk within one operation is allowed.

Two operations share a MAC key only if they carry the same transactionID, because each operation encapsulates afresh and the transactionID enters the key derivation. A key cannot be moved from one operation to another.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value medium: The documents say the transactionID ensures domain separation of the derived key between operations, and that a fresh encapsulation is required per operation.
- Stated in the documents: yes. Where the documents say it: s0082, s0097
- Rests on: ideal KDF: different inputs give different keys; the model gives Bob a fresh transactionID per operation, as a client that uses at most one transaction with a given ID would
- Next step: None.

R01: fresh KEM evidence and nonces

The nonces and KEM encapsulation evidence Bob accepts were freshly generated for this management operation; old evidence cannot serve as valid authentication for it.

The nonce and the encapsulation evidence Bob accepts with are the ones he generated for this operation, so evidence from an earlier operation does not serve as authentication now. The same holds when the operation carries two protected messages.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value medium: The documents say the nonces protect the message against replay and require a fresh encapsulation per operation.
- Stated in the documents: yes. Where the documents say it: s0014, s0097
- Rests on: two protected messages per operation in the unrolled model; fresh randomness for ct and for the nonce
- Next step: None.

R02: no replay within an operation

Within a single management operation, Bob accepts the same logical message at most once; a replayed old message is rejected because the nonce does not match.

Across operations a replayed message fails, because Bob's senderNonce and the shared secret are new for every operation. Inside one operation, a recorded message replayed as the next message is accepted again, because every message Alice sends carries the same recipNonce and the MAC key is reused, so no nonce mismatch can be detected.

- Result: Fails (1 attack); answers what is asked: yes; usable with caveats
- Value medium: The documents say the nonces protect against replay, but they do not state that a message repeated inside an operation is accepted, nor that the recipient must record accepted senderNonces.
- Stated in the documents: unclear. Source text it rests on: s0014
- Rests on: the operation is unrolled to two protected messages; the model keeps no record of accepted senderNonces, because the source requires none; the PKIBody is opaque, so the repeated action is not named
- Open: the consequence of the repeat depends on the message type
- Next step: Model the message types of an operation to name the action that is repeated.

U01: no ss/ct reuse across operations

ss and ct are not reused between different management operations; Bob rejects messages that belong to another transaction or that he has already accepted.

Every operation gets a fresh ciphertext, shared secret and MAC key, and a message whose transactionID is not Bob's own is refused. Neither ss nor ct is reused between operations, and a message of another transaction does not verify.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value medium: The documents require a fresh encapsulation per operation and forbid reuse of ss and ct.
- Stated in the documents: yes. Where the documents say it: s0064, s0097
- Rests on: ideal KDF and MAC; fresh randomness per operation, as the source requires
- Next step: None.

X01: forward secrecy of the MAC key

An attacker who obtains Alice's KEM private key only after a PKI management operation has finished cannot derive that operation's ssk, i.e. the recorded ciphertext and protected message do not give away the past key.

The scheme has no forward secrecy for the MAC key: an attacker who obtains Alice's KEM private key after an operation decapsulates the recorded ciphertext and re-derives the MAC key from the public transactionID and kemContext, so it can verify and forge the messages of that operation. The documents say forward secrecy is not needed for CMP message protection, so this is the expected result.

- Result: Attack holds (stated in the documents); answers what is asked: yes; usable
- Value low: Section 8.5 of the CMP document says perfect forward secrecy is not needed for message protection, and RFC 9629 states that a KEM private key compromise may disclose all contents protected with it.
- Stated in the documents: yes. Where the documents say it: main:3875, rfc9629:568
- Rests on: the attacker is given the KEM private key after the operation, in a later phase of the model; decapsulation with the private key is exact in the symbolic model
- Next step: None.

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, each with the analyst's verdict.

target	level	bound	lemma	control	attacks	claims
A03	proved		a03	checked		1 proved
A04	proved		a04	checked	a04_b (holds_in_source); a04_inj (holds_in_source)	1 proved, 2 attack
D01	proved		d01	checked		1 proved
K01	proved		k01_ss	checked	x01_ss (holds_in_source); x01_ssk (holds_in_source)	3 proved, 2 attack
K04	proved		k04	checked		1 proved
R02	proved		r02	checked	r02_b (holds_in_source)	1 proved, 1 attack
R01	proved		r01	checked		1 proved
A06	proved		a06	checked		1 proved
A08	proved		a08	checked		1 proved
A03.10	proved		a03	checked		1 proved
D01.11	proved		d01	checked		1 proved
U01	proved		u01	checked		2 proved
X01	none				x01_ssk (holds_in_source)	1 attack

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

target	claim	attacker	answer	query	control
A03	Every successful MAC verification by Bob corresponds to a real Alice run with the same transactionID, senderNonce, recipNonce, kemContext and PKIBody (roles and identities as in the flow, Alice holding the KEM private key).	`network`	proved	a03	checked
A04	Each acceptance by Bob corresponds to a different Alice run: the same protected message is not accepted twice.	`network`	proved	a04	checked
A04	The same protected message is not accepted twice by Bob within one management operation.	`network`	attack (holds_in_source)	a04_b	
A04	Within a two-message management operation every acceptance by Bob corresponds to a different send by Alice (injective agreement on the unrolled model).	`network`	attack (holds_in_source)	a04_inj	
D01	PKIHeader fields, PKIBody, transactionID, senderNonce/recipNonce and kemContext of an accepted message are those Alice sent, unmodified.	`network`	proved	d01	checked
K01	ss of an operation stays secret from the attacker on the network.	`network`	proved	k01_ss	checked
K01	ssk derived by Bob stays secret from the attacker on the network.	`network`	proved	k01_ssk	checked
K01	Against an attacker who has Alice's KEM private key after the operation, ss of the recorded operation is recoverable from the public ciphertext.	`kem_sk_leak`	attack (holds_in_source)	x01_ss	

target	claim	attacker	answer	query	control
K01	Against an attacker who has Alice's KEM private key after the operation, ssk of the recorded operation is recoverable from the public ciphertext, transactionID and kemContext.	`kem_sk_leak`	attack (holds_in_source)	x01_ssk	
K01	Alice's KEM private key sk stays secret from the attacker on the network.	`network`	proved	k01_sk	checked
K04	ssk values of different management operations differ (the same ssk occurs only under the same transactionID).	`network`	proved	k04	checked
R02	A replayed protected message is not accepted by Bob: acceptance is injective in Alice's run and bound to Bob's fresh nonce.	`network`	proved	r02	checked
R02	Within one management operation Bob accepts a given protected message at most once; a replayed old message is rejected because its nonce does not match.	`network`	attack (holds_in_source)	r02_b	
R01	The nonce and KEM encapsulation evidence Bob accepts with are the ones he produced for this operation.	`network`	proved	r01	checked
A06	Bob's successful MAC verification confirms that Alice derived and holds the same ssk for this operation and context.	`network`	proved	a06	checked
A08	The transactionID and kemContext under which Bob accepts are the ones of his own encapsulation for this operation.	`network`	proved	a08	checked
A03.10	Bob's successful verification implies a run of the party holding Alice's KEM private key, with matching transactionID and nonces.	`network`	proved	a03	checked
D01.11	An accepted protected message has PKIHeader, PKIBody, transactionID and nonces corresponding to Alice's real send.	`network`	proved	d01	checked
U01	Bob rejects a protected message belonging to another transaction: the transactionID of an accepted message is the one of his own encapsulation.	`network`	proved	u01	checked
U01	ss and ct are not reused between management operations: two operations share an ssk only if they carry the same transactionID.	`network`	proved	k04	checked
X01	An attacker who obtains Alice's KEM private key only after the operation has finished cannot derive the ssk of that operation.	`kem_sk_leak`	attack (holds_in_source)	x01_ssk	

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-20261114T230140, 2026-11-14. Status: finished. Model: deepseek-v4-1-flash-261017. Steps: 41, prover 5 s, wall 10 min, cost 0.082 USD.

ProVerif: 106 proofs run, 106 decided, 0 without an answer, 4 s, 27 answers recorded.

What was analysed

Protocol	CMP KEM-based MAC protection
Version	rfc9810
Analysed	Selected generic KEM-based MAC message-protection flow, where Alice has an authenticated ML-KEM public key, Bob generates a fresh ss/ct, Alice MACs the PKIMessage and Bob verifies it; transactionID, senderNonce/recipNonce, roles and kemContext are modeled explicitly. This is not full certificate issuance or RA registration validation.
Not analysed	The symbolic model does not prove computational security or quantum resistance of the algorithms; attack traces hold only under explicit assumptions and are not directly equivalent to specification flaws. Every property must have a reachability/positive control.
Source text	100 numbered passages (23041 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	5.1.1	PKI Message Header
main	5.1.1.1	ImplicitConfirm
main	5.1.1.2	ConfirmWaitTime
main	5.1.1.3	OrigPKIMessage
main	5.1.1.4	CertProfile
main	5.1.1.5	KemCiphertextInfo
main	5.1.3.4	Key Encapsulation
main	5.3.19.18	KEM Ciphertext
main	8.8	Recurring Usage of KEM Keys for Message Protection

The purpose of the protocol as the source step located it in the documents: Selected generic KEM-based MAC message-protection flow, where Alice has an authenticated ML-KEM public key, Bob generates a fresh ss/ct, Alice MACs the PKIMessage and Bob verifies it; transactionID, senderNonce/recipNonce, roles and kemContext are modeled explicitly. This is not full certificate issuance or RA registration validation.

The properties against the documents

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

ID	Name	About	What the documents say	Where
A03	Bob authenticates Alice	Bob is assured, when the MAC verifies, that Alice (the holder of the KEM private key sk) ran the protocol for this management operation with the same transactionID, nonces and kemContext.	The flow ends with 'Alice authenticated by Bob'; Bob verifies the MAC with the shared secret key only Alice can derive from the ciphertext.	s0047, s0061, s0074
A04	distinct Alice runs per acceptance	Every successful acceptance by Bob corresponds to a different Alice run; no protected message is accepted twice.	Bob MUST NOT reuse ss and ct for other operations, and a fresh Encapsulate() call is required per operation.	s0014, s0064, s0097
D01	MAC binds the message fields	The PKIHeader fields, PKIBody, transactionID, nonces and kemContext of an accepted message are those Alice sent, unmodified.	If a PKI message is protected then that information is also protected; PKIProtection carries a MAC over the message.	s0002, s0048, s0074, s0080, s0081, s0082, s0083
K01	KEM secrets stay secret	ss, ssk and Alice's KEM private key sk are kept from the attacker who sees the public ciphertext, the protected message and the public key.	The KEM shared secret comes out of Decapsulate with the private key only; MAC-based protection relies on the derived ssk.	s0044, s0045, s0046, s0069, s0072, s0099
K04	key separation between operations	ssk values derived in different PKI management operations are different, so an ssk is confined to its own operation.	transactionID is used to ensure domain separation of the derived shared secret key between different operations, and a fresh ss/ct is required per operation.	s0064, s0075, s0082, s0083, s0097
R02	no replay within an operation	Within one management operation Bob accepts a given protected message at most once; an old message is rejected because its nonce does not match.	The recipNonce is copied from the senderNonce of the previous message in the transaction; the nonces protect the PKIMessage against replay attacks.	s0014
R01	fresh KEM evidence and nonces	The nonce and the KEM encapsulation evidence Bob accepts with belong to this operation, not to an older one.	senderNonce and recipNonce protect the PKIMessage against replay; a fresh Encapsulate() per operation.	s0014, s0064, s0097
A06	key confirmation to Bob	Bob's successful verification confirms that Alice derived and holds the same ssk for this operation and context.	Bob uses the shared secret key for verifying the MAC-based protection of the message received and in this way authenticates Alice.	s0061, s0074, s0075
A08	operation context consistency	The transactionID and kemContext under which Bob accepts a message are the ones of his own encapsulation for this operation.	transactionID MUST be the value from the message containing the ciphertext, and is used for domain separation; kemContext is algorithm-specific context shared by both sides.	s0012, s0013, s0080, s0081, s0082, s0083
A03.10	Bob authenticates the KEM key holder	Bob's successful MAC verification corresponds to a run of the party holding Alice's KEM private key, with matching transactionID and nonces (user question Q1).	Alice is defined as the entity possessing the KEM key pair who provides MAC-based protection; Bob verifies it.	s0047, s0061, s0074
D01.11	header and body binding	An accepted protected message has PKIHeader, PKIBody, transactionID and nonces corresponding to Alice's real send (user question Q2).	The protection covers the whole PKIMessage; transactionID is used for domain separation of the derived key.	s0002, s0048, s0074, s0082

ID	Name	About	What the documents say	Where
U01	no ss/ct reuse across operations	ss and ct are not reused between management operations and Bob rejects messages belonging to another transaction or to one already accepted (user question Q3).	Bob MUST NOT reuse ss and ct for other operations; a server rejects a transactionID already in use.	s0013, s0014, s0064, s0097
X01	forward secrecy of the MAC key	the ssk of an already completed operation, against an attacker who later obtains Alice's KEM private key sk	Section 8.5 says perfect forward secrecy is not needed for CMP message protection providing integrity and authenticity, and RFC 9629 says compromise of the KEM private key may result in the disclosure of all contents protected with that key.	main:3875, rfc9629:568

Added from the documents by the analyst: X01.

The security statements of the documents, paragraph by paragraph

The host listed the 74 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: 6. Not taken: 68. 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:3800-3812	Outside the scope	POP for certificate issuance; the task models only the KEM-based MAC message-protection flow.
P02	main:3816-3830	Outside the scope	POP by decryption of the certificate; no certificate issuance in the modelled flow.
P03	main:3832-3835	Outside the scope	Rules for returning decrypted POP challenges; not part of message protection.
P04 to P07	main:3839-3852	Outside the scope	POP by exposing the private key to the CA/RA; no key archival or POP in the modelled flow.
P08	main:3856-3871	Outside the scope	Small-subgroup attack on a DH key exchange in key archival/recovery; the modelled flow uses a KEM, not DH, and no key recovery.
P09	main:3875-3883	X01, `kem_sk_leak`	
P10	main:3885-3889	Outside the scope	PFS for CMS EnvelopedData (encrypted values) and TLS transport; outside the MAC-protection flow.
P11	main:3893-3896	Outside the scope	Separation of a CA certificate-signing key from CMP message-protection keys; the modelled flow has no CA signing key.
P12	main:3899-3899	No goal stated	Section heading, states no property.
P13	main:3901-3912	The prover cannot be asked	Quality of the PRNG used to generate nonces and private keys; a computational/implementation property.
P14	main:3914-3923	The prover cannot be asked	Entropy of shared secret information, passwords as shared secrets; the KEM produces a high-entropy ss, and guessing entropy is a computational property.
P15	main:3925-3930	The prover cannot be asked	Required entropy of shared secret information relative to delivered key strength; a computational requirement.
P16	main:3932-3943	Outside the scope	Entropy requirements when delivering trust anchors; trust anchor provisioning is outside the modelled flow.
P17	main:3945-3947	No goal stated	Pointer to the algorithm profiles of RFC 9481, states no property.
P18	main:3951-3954	K04, U01	

Paragraphs	Where	Taken up as	Why not
P19	main:3956-3958	The prover cannot be asked	Assumption that the overall data size protected by one shared secret key is small; a volume/size assumption, not a trace property.
P20	main:3960-3969	The prover cannot be asked	Requirement that the KEM be IND-CCA2 secure under public-key reuse; strength of a cryptographic primitive.
P21	main:3971-3973	The prover cannot be asked	Consequence of the IND-CCA2 assumption for unlimited public-key reuse; a computational statement.
P22 to P24	main:3977-3995	Outside the scope	Trust anchor provisioning with CMP messages; outside the modelled flow.
P25	main:3999-4004	Outside the scope	Authorization for certificates with specific EKUs; certificate issuance policy, outside the modelled flow.
P26 to P27	main:4008-4018	Outside the scope	CT log publication and revocation; outside the modelled flow.
P28 to P52	rfc5280:5226-5449	Outside the scope	RFC 5280 certificate and path validation security considerations; certificate validation is not part of the modelled flow.
P53	rfc9481:1050-1056	The prover cannot be asked	Security level of the chosen algorithms; strength of primitives.
P54	rfc9481:1058-1074	The prover cannot be asked	Choice between PBMAC1 and PasswordBasedMac for password-based protection; algorithm choice, not the KEM flow.
P55	rfc9481:1076-1078	The prover cannot be asked	Prohibition on using AES-GMAC as the PBKDF2 PRF; a primitive-level requirement.
P56	rfc9481:1080-1092	No goal stated	Migration advice on algorithm profiles, states no property of this protocol.
P57	rfc9481:1094-1098	The prover cannot be asked	Choosing a suitable PRF and sufficient key entropy for symmetric MAC algorithms; primitive strength.
P58	rfc9629:542-543	No goal stated	Makes the security considerations of RFC 5652 applicable; states nothing of this protocol itself.
P59	rfc9629:545-554	The prover cannot be asked	Requirement that the KEM be secure under many-time public-key use (IND-CCA2); primitive strength.
P60	rfc9629:556-556	The prover cannot be asked	KDF should offer at least the security level of the KEM; primitive strength.
P61	rfc9629:558-561	The prover cannot be asked	Key-encryption algorithm and KEK sizes; primitive strength, and not used in the MAC-based flow.
P62	rfc9629:563-566	A03, D01	
P63	rfc9629:568-572	K01, X01, `kem_sk_leak`	
P64	rfc9629:574-587	K04, U01	
P65	rfc9629:589-598	The prover cannot be asked	Random generation of keys and PRNG quality; computational/implementation property.
P66 to P68	rfc9629:600-620	The prover cannot be asked	Effective security is the weaker of two algorithm sizes; primitive strength.
P69	rfc9629:622-629	No goal stated	Advice that algorithms weaken over time and implementations should be modular; states no property of the protocol.
P70	rfc9935:406-407	Outside the scope	RFC 5280 security considerations; certificate validation is outside the modelled flow.
P71	rfc9935:409-411	K01, `kem_sk_leak`	

Paragraphs	Where	Taken up as	Why not
P72	rfc9935:413-421	The prover cannot be asked	PRNG quality for private key generation; computational property.
P73	rfc9935:423-431	The prover cannot be asked	Binding properties (LEAK-BIND, MAL-BIND) of ML-KEM private key formats; a computational property of the KEM.
P74	rfc9935:433-435	No goal stated	Pointer to detailed ML-KEM security considerations, states no property.

The parties and the protected values, one by one

The host listed the 18 parties and protected values that the drafted properties name, and the analyst said of each which attacker controls the party or holds the value, or why no such attacker was set. With an attacker: 1. Without: 17. 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	Alice	party	Everything lost by definition	An attacker controlling Alice holds the KEM private key and can produce every message Alice could produce, so Bob's authentication of Alice is satisfied by the attacker itself; no other target is asked about it.
S02	Bob	party	Nothing gained	Bob holds only values it generated itself (ss, ssk) and Alice's public key; controlling Bob gives no secret of Alice and no property of the targets is about Bob's honesty.
S03	Attacker	party	Nothing gained	The network attacker is already the baseline attacker.
S04	ssk	value	Everything lost by definition	Handing the ssk of an operation to the attacker makes forgery in that very operation possible by definition; no target asks about anything else that ssk protects.
S05	ss	value	Everything lost by definition	ss determines ssk through the public KDF, so giving ss away loses exactly the secrecy K01 asks for that operation.
S06	ct	value	Nothing gained	ct is sent in the clear to Alice, so the attacker on the network already has it.
S07	transactionID	value	Nothing gained	transactionID travels in the clear in the header and the ciphertext message.
S08	senderNonce/recipNonce	value	Nothing gained	senderNonce and recipNonce travel in the clear in the header.
S09	kemContext	value	Nothing gained	kemContext is transferred in the protection parameters and is public context data.
S10	senderNonce	value	Nothing gained	senderNonce is public header data.
S11	recipNonce	value	Nothing gained	recipNonce is public header data.
S12	PKIHeader fields	value	Nothing gained	PKIHeader fields are sent in the clear; MAC-based protection gives integrity, not confidentiality.
S13	PKIBody	value	Nothing gained	PKIBody is not encrypted by MAC-based protection; only its integrity is at stake, which D01 asks about.
S14	MAC value	value	Nothing gained	The MAC value is public; knowing it gives no key.
S15	Alice's KEM private key sk	value	`kem_sk_leak`	
S16	kdf/len/mac parameters	value	Nothing gained	kdf, len and mac parameters are transferred in the protectionAlg and are public.
S17	staticString	value	Nothing gained	staticString is the fixed public constant "CMP-KEM".
S18	PKIHeader	value	Nothing gained	The PKIHeader is sent in the clear; its integrity is what D01 asks about.

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		
`kem_sk_leak`	Alice's KEM private key sk, obtained after the operation it is used against (for the secrecy of ss and ssk this is enough because the ciphertext ct is public)	RFC 9629 says implementations MUST protect the KEM private key and that compromise of the KEM private key may result in the disclosure of all contents protected with that key; RFC 9935 says disclosure of private key material can lead to masquerades.	rfc9629:568, rfc9935:409

ID	`network`	`kem_sk_leak`
A03	Holds	
A04	Fails	
D01	Holds	
K01	Holds	Fails
K04	Holds	
R02	Fails	
R01	Holds	
A06	Holds	
A08	Holds	
A03.10	Holds	
D01.11	Holds	
U01	Holds	
X01		Fails

Source and targets

Source: 100 units, 23041 bytes, frozen in source-delivery-01.json. Targets: 10 approved (A03, A04, D01, K01, K04, R02, R01, A06, A08, U01); the questions asked: 3. The list of targets was approved as drafted, beforehand, with --approve-targets.

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

- approve_package: approved source package protocol-3a2c98f241f72293e2214c14eb4c

This step as it reports itself: 1 requests (1 failed), 0.000 USD, 22983 tokens in and 4281 out, 28 s.

Notes on the prover and the model

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

- R02: 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.

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
Bob's expected recipNonce for messages of the operation (his own senderNonce nb)	Bob	when Bob sends a new PKIMessage of the transaction; in this flow Bob sends only the KemCiphertextInfo message, so the expected recipNonce stays nb for every message Alice sends in the operation	on every protected message received from Alice, which must carry recipNonce = nb	s0014, s0068
senderNonce of each protected message sent by Alice	Alice	for every protected PKIMessage Alice sends (fresh 128-bit random value per message)	only indirectly: Bob copies it into recipNonce when he sends the next message of the transaction; Bob does not compare it against anything he remembers	s0014
ssk of the management operation	Alice and Bob	never inside the operation; a new operation invokes Encapsulate again and yields a new ct, ss and therefore a new ssk	on every protected message received, by verifying the MAC under ssk	s0064, s0075, s0097
record of senderNonces already accepted inside the operation	no party: the source states no such record or counter	not applicable	not applicable	s0014

pattern	result	target	shown by ProVerif	steps
replay	candidate	R02	yes: ProVerif falsifies `r02_b` on bounded.pv	Bob invokes Encapsulate(pk_A) for the operation, derives ss and ssk and sends the KemCiphertextInfo message with transactionID, his senderNonce nb and kemContext (s0064, s0090, s0091). / Alice decapsulates ct, derives ssk and sends a protected PKIMessage with senderNonce na1 and recipNonce nb (s0066..s0074, s0014). / Bob checks recipNonce = nb and the MAC under ssk and accepts the message (s0014, s0074). / Alice sends the next protected message of the same operation with a fresh senderNonce na2, again recipNonce nb and the same ssk (s0014, s0075). / The attacker withholds that next message and replays the first protected message instead. / Bob checks recipNonce = nb, which still matches because his senderNonce did not change, and the MAC, which still verifies under the reused ssk, so he accepts the earlier message a second time (s0014, s0074, s0075).

pattern	result	target	shown by ProVerif	steps
repeat_one_time_action	candidate	A04	yes: ProVerif falsifies `a04_inj` on bounded.pv	Bob sends the KemCiphertextInfo message with transactionID, senderNonce nb and kemContext and derives ssk (s0064). / Alice sends a first protected message, senderNonce na1, recipNonce nb, MAC under ssk; Bob accepts it (s0014, s0074). / The attacker intercepts a later protected message of the same operation and replays the first one instead. / Bob accepts the same protected message again, so whatever action its PKIBody requests is carried out twice for one send by Alice (s0074, s0075).
desynchronisation	nothing found			Bob's expected recipNonce only changes when Bob sends a message, and the ssk is fixed for the operation; there is no second Bob message in this flow that Alice could miss, and both parties derive ssk from the same ss and transactionID (s0082), so the two sides cannot be brought to different keys. model.pv verified A08.
update_before_check	nothing found			No kept value is changed on receipt of a message: the ssk and the expected recipNonce are fixed before any protected message arrives and are only compared (s0014, s0074).
no_rollback_on_failure	nothing found			A failed decapsulation or MAC check terminates the operation (s0068) and no kept value is modified, so nothing is left changed; the abandoned operation's ct and ssk are simply discarded (s0064).
race_between_sessions	nothing found			Every operation has its own fresh ct, ss, ssk and transactionID (s0064, s0083, s0097), so no value is shared between two operations. model.pv verified that two operations share an ssk only under the same transactionID (K04).
distinguishable_failure	nothing found			The source prescribes a single failInfo value, badMessageCheck, for a failed decapsulation (s0068), so no distinct failure answers are specified; the model has no error response and no event that tells the attacker which check failed.
wrap_or_window	nothing found			The nonces are 128-bit random values compared only for equality with Bob's own value (s0014); they are not counters and no acceptance window is specified, so neither wrap-around nor a window applies.

What the analyst named for targets:

step	targets	features	reason
20	A04, R02	changing_state	A04 and R02 ask that a protected message not be accepted twice, which depends on how many messages one management operation carries and on what the recipient has already accepted in that operation. model.pv has one acceptance per operation and leaves out that per-operation record, so both need the unrolled model.
20	A03, A03.10, D01, D01.11, A06, A08, K01, K04, R01, U01, X01	plain_primitives	The remaining targets rest on the KEM, the KDF and the MAC as ideal symbolic primitives (a one-way encapsulation with a matching decapsulation, a one-way key derivation, an unforgeable MAC). No algebraic law of these primitives is used, but no computational property of them is proved either.

The analyst's closing note

Analysed the generic CMP KEM-based MAC-protection flow (RFC 9810 5.1.3.4, 5.3.19.18, 8.8, 5.1.1) in ProVerif. A model of the plain run was built first (all four send/accept events reachable); all 13 targets then got answers, most of them proofs over any number of sessions.

Proved against the network attacker: Bob's acceptance implies a real Alice run with the same transactionID, senderNonce, recipNonce, kemContext and body (A03, A03.10); injective agreement across operations (A04); the MAC binds every header field and the body (D01, D01.11); the successful MAC verification is key confirmation (A06); the operation context is Bob's own (A08); the nonce and encapsulation evidence are fresh (R01); the shared secret, the MAC key and Alice's KEM private key stay secret (K01); keys are separated between operations (K04, U01). All proofs have falsified controls.

One new weakness: inside one management operation the MAC key is reused and every message from Alice carries the same recipNonce, so a recorded protected message replayed as the next message of the same operation passes both checks and is accepted a second time (A04, R02). The documents claim replay protection for the nonces but never require the recipient to record accepted senderNonces. A recipient-side nonce record removes the attack; the change was checked on the unrolled model and leaves every other answer as it is.

Reproduced as documented, not as findings: no forward secrecy of the MAC key against a later KEM private-key compromise (added target X01; main 8.5, RFC 9629 section 7), and recovery of ss and ssk by the holder of Alice's KEM private key (K01 under kem_sk_leak).

Blocks: the KEM, KDF and MAC are ideal symbolic primitives; the replay result rests on an operation unrolled to two messages with an opaque PKIBody; one sender-recipient pair and one direction only. Tamarin+ Agent, which models the primitives and the message types as they are, would settle whether a repeated message of a concrete CMP type has a harmful effect.