IEEE P802.11  
Wireless LANs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Erratta Comments on SAE | | | | |
| Date: 2024-01-05 | | | | |
| Author(s): | | | | |
| Name | Affiliation | Address | Phone | email |
| Dan Harkins | HPE |  |  |  |
|  |  |  |  |  |

Abstract

Some issues with the definition of SAE in IEEE Std 802.11-2020 were reported as errata to the Chair of IEEE 802.11. They are being treated as comments and are addressed in this submission.

*First Erratum [Password identifier]*

*Location:* §12.4.5.4, pg. 2463.

*Original Text: If the peer’s SAE Commit message con- tains a password identifier, the value of that identifier shall be used in construction of the password element (PWE) for this exchange*

*Correction:* Complete with further instructions, consid- ering Mesh connections (which is included as part of the standard: ”SAE shall be implemented on all mesh STAs to facilitate and promote interoperability.“) and what behaviour should be followed by participants that both send a *different* password identifier to their peer.

*Rationale:* It is not clear what exact operations the receiver is supposed to perform (as this is underspecified in the standard); but we guess that it should at least change EA (and made this assumption when modelling this), however this should be made explicit. In particular, the scenario of what happens if each party sends a *different* password identifier to their peer is left unexplained. We modelled this scenario where both follow the current specs (interpreted as updating EA) with ProVerif (a state-of-the-art symbolic verification tool) and can mimic a deadlock situation, results show this underspecification may lead to a DoS attack.

Discussion: There is a 1:1 binding between a password identifier and a password. While discrete passwords can have the same value and be identified by different identifiers they are treated as different (even though the value of the password is identical). There is only 1 password used in a single SAE exchange and therefore the same identifier will identify it on both sides. The state machine mentions that a bad match on password identifiers in Commit messages results in a failure of the protocol.

Proposed Resolution: Reject.

*Second Erratum [List of rejected groups]*

*1) Location:* §12.4.5.4, pg. 2463; §12.4.7.4, pg. 2467.

*2) Original Text: In the first message, a list of previously rejected group elements is sent. However, if the list contains a group element that wouldn’t be rejected, then the protocol aborts. This is different from SAE’16 in that only the last rejected group is considered. Another insight provided by the standard is “If an SAE Commit message [...] is being sent in response to rejection of a previous SAE Commit message with status code set to UNSUPPORTED\_FINITE\_CYCLIC\_GROUP”, that vaguely suggests that after a rejection a new commit message is sent.*

*3) Correction:* Keep a list of proposed groups, then drop the communication if among the received rejection list is not coherent with the list of proposed groups.

*4) Rationale:* It remains unspecified what the other fields should contain, e.g., if they can offer another group or all other fields are mainly ignored, which leaves an ambiguous decision for the implementation.

One possible interpretation is that after a rejection, the peer always sends a new commit negotiating another group. We can see a probably unwanted behaviour that can become a vulnerability if an attacker has advantages of a specific target group. In particular, if you send a message with a group that would be accepted, e.g., 25, an attacker may reply with a forged rejection message; this triggers the peer to use another group, e.g., 21, and appends 25 in the list of rejected groups. However, as this is sent in plaintext, the attacker would substitute 25 with a group that would be rejected, e.g., 19, and the receiver might accept group 21. Overall, this would allow an attacker to force the group negotiation to a specific target group. We modelled this behaviour in ProVerif and see that a specific group can be forced and, thus, a potential vulnerability may be introduced.

Discussion: Yes, a new Commit message is sent upon receipt of a rejection for unsupported group. A list of proposed groups is retained in the form of the Rejected Group Element and that list is checked each time a commit message is received to ensure that there are no groups there that would not have been rejected. Yes, an attacker can continue to rewrite a frame to remove/replace this group and the commit message will be processed but since the rejected groups are added to the KDF by each side, any modification of rejected groups by an attacker would result in generating different keys, the subsequent Confirm messages would be rejected, and the protocol would fail. While this may not be the most graceful of failures, the attack is highly contrived and ultimately is not successful anyway.

Proposed Resolution: Reject

*Third Erratum [Start with different groups]*

*1) Location:* §12.4.5.4, pg. 2463.

*2) Original Text: Peers are allowed to both start the protocol simultaneously but with a different group.*

*3) Correction:* Restart with a random delay (like it’s done in networks that share a single bus), and pick a *random* group for the negotiation until they get rejections or simply match. There are so few groups, that worrying about efficiency would be unjustified: for example, the average number of attempts to expect a match is n2 with n possible groups to chose from. Additionally, randomizing the delay for restarting significantly decreases the likelihood that both peers would start simultaneously again, dropping the number of attempts to successfully start the protocol to just 2.

*4) Rationale:* No instructions are provided on what to do in this case. Group elements can get rejected, but it is not the case here. We are in the case where a peer just starts the protocol with a different group than another one (problematic in mesh networks).

Discussion: This is discussed in the state machine §12.4.8.6.4 (Protocol Instance in Committed state) when the DiffGrp indicator is set. While it would work to restart with a random delay, the existing description is deterministic and straightforward—the peer with the numerically greater MAC address “wins” and its chosen group is used in the exchange—without requiring any additional exchanges, save the retransmission of the winner’s Commit message so the number of attempts to successfully start the protocol is just 1.

Proposed Resolution: Reject.

*Fourth Erratum [Principal role]*

*1) Location:* §12.4.1, pg. 2452; §12.4.4.2.2, pg. 2455; §12.4.4.3.2, pg. 2460; §12.4.4.3.3, pg. 2461;

*2) Original Text: Unlike other authentication protocols SAE does not have a notion of an “Initiator” and “Responder” or of a “Supplicant” and “Authenticator.”*

*3) Correction:* Occurrences of ’SAE Initiator’ should be removed, being careful to consider the impact this will have on the fields of the commit message.

*4) Rationale:* The SAE protocol does not define roles and assuming that there is an agent initiating the exchange (§12.4.4.2.2, §12.4.4.3.2, §12.4.4.3.3) it allows for the possibility of simultaneous sending of commit messages, which is expected by the protocol.

Discussion: The assumptions on SAE in §12.4.2 state that “Each side may initiate the protocol simultaneously such that each side views itself as the ‘initiator’ for a particular run of the protocol.” So yes, there is no strict role, but the protocol is initiated by an SAE peer and that peer can view itself as the party that initiated the particular run even if the other side does as well.

Proposed Resolution: Revise, instruct the editor to make the following changes to 12.4.2:

**12.4.2 Assumptions on SAE**

Unlike other authentication protocols SAE does not -force peers into strict roles where one is an ‘Initiator’ and the other is the ‘Responder’ or one is ‘Supplicant’ and the other ‘Authenticator.’ The parties to the exchange are equals, with each side being able to initiate the protocol. Each side may initiate the protocol simultaneously such that each side views itself as the ‘initiator’ for a particular run of the protocol. This is necessary to address the unique nature of MBSSs.

*Fifth Erratum [Missing case when using the Anti-Clogging Token]*

*1) Original Text: As long as the state machine variable Open is greater than or equal to dot11RSNASAEAntiCloggingThreshold all SAE Commit messages that do not include a valid Anti-Clogging Token field shall be rejected with a request to repeat the SAE Commit message and include the token (see 12.4.5.1).*

*2) Location:* §12.4.6, pg. 2465.

*3) Correction:* On page 2471 the text should be amended to *’greater or equal’* in the sentence: ’If Open is *greater* than dot11RSNASAEAntiCloggingThreshold, the parent process shall check for the presence of an Anti-Clogging Token field.’ instead of just ‘greater‘.

*4) Rationale:* The protocol requires the use of the token as if Open is equal to dot11RSNASAEAntiCloggingThreshold, but the case when: open is equal to dot11RSNASAEAntiCloggingThreshold and the generated Anti-Clogging token is correct is not provided by the protocol. The protocol is executed with multi- agent logic, which involves simultaneous deletion and creation of instances. Therefore, it is possible for a protocol instance to have a valid Anti-Clogging token available to it even when the Open variable is equal to dot11RSNASAEAntiCloggingThreshold.

Discussion: Yes, this is correct.

Proposed Resolution: Accept.

*Sixth Erratum [Incomplete KDF formula]*

*1) Location:* §12.4.5.4, pg. 2464.

*2) Original Text: If both SAE Commit messages indicated a status code of SAE\_HASH\_TO\_ELEMENT, a salt consisting of the concatenation of the rejected groups from each peer’s Rejected Groups element shall be passed to the KDF*

*3) Correction:* Reflect this modification to the pseudocode below.

*4) Rationale:* This change is not included in the pseudocode below.

Discussion: The salt is included in the derivation of keyseed and the description says the salt “is either a series of 0 octets or a list of rejected groups (see 12.4.7.4 (Encoding and decoding of SAE Commit messages))”. The pseudocode and description already include this information.

Proposed Resolution: Reject.

*Seventh Erratum [Title not in line with the principal description]*

*1) Location:* §12.4.8.3.1, pg. 2469; §12.4.8.3.2 pg. 2469.

*2) Original Text: Parent process events and output’ and ’Protocol instance events and output*

*3) Correction:* Events should be placed in their own separate section, as it stands the section header verbiage is misleading.

*4) Rationale:* Section titles seem to suggest that the reader should find descriptions of events and outputs produced by the participant mentioned in the title. Instead, in the paragraphs, there are events sent by other participants mixed with output produced by the participant to which the section refers.

Discussion: These processes receive events and produce outputs, that is exactly what is being described in the section. The events “sent by other participants” are events processed by the particular process so it makes sense to describe them in a section on the process’s events and outputs.

Proposed Resolution: Reject.

*Eigth Erratum [Conflation of Participants]*

*1) Location:* §12.4.8.3.1, pg. 2469;

*2) Original Text: The SME signals the following events to the parent SAE process: — Initiate. An Initiate event is used to instantiate a protocol instance to begin SAE with a designated peer. — Kill. A Kill event is used to remove a protocol instance with a designated peer.*

*3) Correction:* A section should be created for the SME participant, to clarify its role in the protocol, just like for the other participants.

*4) Rationale:* The SME is a participant that is part of the protocol, just like the Parent process and protocol instances, so it is important to separate it from the Parent process logic. This will make it easier to add new events or outputs to the SME, as well as make the specification more semantically correct.

Discussion: The SME exists outside of SAE while the Parent process and Protocol Instance are defined by SAE. Therefore what is being described in the SAE state machine is how an SAE-defined entity (the Parent Process) interacts with the SME, it is not intended to be a description of the SME.

Proposed Resolution: Reject.

*Ninth Erratum [Clarification on the use of indicators] 1) Location:* §12.4.8.5.2, pg. 2470;

*2) Original Text: In addition, protocol instances maintain the following six indicators that are not maintained as state variables but, instead, indicate the cause of certain behaviour.*

*3) Correction:* Describe why the indicators were introduced or consider removing them.

*4) Rationale:* Indicators are only used to indicate errors. Typically, once an indicator is set, the protocol instance is terminated right away. Indicators are useful for debugging purposes, but it is important to state their intended use clearly, otherwise they lose their usefulness.

Description: They indicate the reason a particular path was taken in the state machine. They can help explain the inner workings of the state machine, and possible help debug errors.

Proposed Resolution: Revised, instruct the editor to modify §12.4.8.5.2 as indicated:

**12.4.8.5.2 Protocol instance variables**

In addition, protocol instances maintain the following six indicators that are not maintained as state variables but, instead, indicate the cause of certain behavior. These indicators can be used to describe the inner workings of the state machine and can be useful for debugging purposes.

*Tenth Erratum [Misleading description of the SME events ]*

*1) Location:* §12.4.8.6.1, pg. 2471;

*2) Original Text: Upon receipt of an Initiate event, the parent process shall check whether there exists a protocol instance for the peer MAC address (from the Init event) in either Committed or Confirmed state.*

and

*Upon receipt of a Kill event, the parent process shall delete all protocol instances indexed by the peer MAC address (from the Kill event) in its database.*

*3) Correction:* The text within the parenthesis ”(from the Init event)” and ”(from the Kill event)” should be removed, or it should be explained in detail why these restrictions are necessary.

*4) Rationale:* The text in the parentheses of the two sen- tences seems to refer to a constraint of the execution flow based on the reception of the *Init* or *Kill* events. However, this is an assumption based on our personal understanding, as the text leaves this unclear. Moreover, the two events are generated by two different principals (Init generated by Parent Process and Kill generated by SME) making the meaning even more complex to grasp.

Discussion: The Initiate event is received by the Parent process when the SME instructs the Parent process to initiate a run of the protocol. The peer to whom the protocol is initiated is identified by MAC address. Similarly, a Kill even is received by the Parent process when the SME requires that all protocol instances of a peer, identified by MAC address, are to be deleted. The text should note that the event identifies the peer by MAC address. Also, the behavioral text is confusing as the MAC address is from the *Initiate* event (from the SME) not the *Init* event (to the Protocol instance).

Proposed Resolution: Instruct the editor to modify §12.4.8.3.1 and §12.4.8.6.1 as indicated:

**12.4.8.3.1 Parent process events and output**

The SME signals the following events to the parent SAE process:

* Initiate. An Initiate event is used to instantiate a protocol instance to begin SAE with a designated peer identified by its MAC address.
* Kill. A Kill event is used to remove a protocol instance with a designated peer identified by its MAC address.

**12.4.8.6.1 Parent process behavior**

Upon receipt of an Initiate event, the parent process shall check whether there exists a protocol instance for the

peer MAC address (from the Initiate event) in either Committed or Confirmed state. If there is, the Initiate event

shall be ignored. Otherwise, a protocol instance shall be created, and an Init event shall be sent to the protocol

instance.

*Eleventh Erratum [Non-existent event]*

*1) Location:* §12.4.8.6.1, pg. 2471;

*2) Original Text: Upon receipt of a Sync, Del, or Fail event from a protocol instance, the parent process shall decrement the Open counter and deletes the protocol instance.*

*3) Correction:* Sync should be replaced by big(Sync)

*4) Rationale:* Sync is not an event (it is a variable). The event associated with exceeding the maximum threshold related to the variable Sync is big(Sync).

Discussion: Yes, this is correct.

Proposed Resolution: Accept.

*Twelveth Erratum [Declared but never used event]*

*1) Location:* §12.4.8.6.1, pg. 2471;

*2) Original Text: Upon receipt of a Sync, Del, or Fail event from a protocol instance, the parent process shall decrement the Open counter and deletes the protocol instance.*

*3) Correction:* Fail should be removed, or a use case scenario should be provided.

*4) Rationale:* Fail is defined as the event produced by the protocol instance in case of an authentication failure. This event is never mentioned in the state machine of the protocol instance. All error instances leading to protocol instance termination are generated from a Del event. Consequently, Fail is never sent. The two events tend to have an overlapping role, and unless a specific use case exists, one of the two is redundant.

Discussion: it’s useful to distinguish between Del and Fail because Fail will happen with a password mismatch and Del can happen for some other reason (e.g. too many sync errors, etc). But the commenters are correct that an indication of Fail is never sent. Looking at figure 12-4 it seems that when the indicator *BadAuth* is set a Fail event should be generated and BadAuth is set when there’s an authentication failure—i.e. when the Commit message is properly parsed but not verified.

Proposed Resolution: Revised, instruct the editor to make the following changes in §12.4.8.6.5 and to the indicated text fragment from figure 12-4 which is on a line from Confirmed state to Nothing state (sorry, no Visio).

**12.4.8 SAE finite state machine**

**12.4.8.1 General**

(Con, BadAuth, !big(sync)/Fail

**12.4.8.6.5 Protocol Instance behavior—Confirmed state**

Upon receipt of a Con event, the SAE Confirm message shall be processed according to 12.4.5.6 (Processing of

a peer’s SAE Confirm message). If processing is unsuccessful and the SAE Confirm message is not verified,

protocol instance shall set BadAuth, send the parent process a Fail event, and transition back to *Nothing* state. If processing is successful and the SAE Confirm message has been verified, the Rc variable shall be set to the value of the Send-Confirm field, Sc shall be set to the value 216 – 1, the t1 (key expiration) timer shall be set, the t0 (retransmission) timer shall be canceled, and the protocol instance shall transition to Accepted state.

*Thirteenth Erratum [Declared but never used event]*

*1) Location:* §12.4.8.6.1, pg. 2471;

*2) Original Text: Upon receipt of an Auth event from a protocol instance, the parent process shall decrement the Open counter.*

*3) Correction:* Auth event should be added to section §12.4.8.6.5 when the protocol instance transition into the Accepted state.

*4) Rationale:* The Auth event is defined, but its use in the state machine is never described. It would be appropriate to add it at the end of a run that successfully reaches the Accepted state.

Discussion: This is correct.

Proposed Resolution: Revised, instruct the editor to modify §12.4.8.6.5 as indicated:

Upon receipt of a Con event, the SAE Confirm message shall be processed according to 12.4.5.6 (Processing of

a peer’s SAE Confirm message). If processing is unsuccessful and the SAE Confirm message is not verified,

protocol instance shall remain in Confirmed state. If processing is successful and the SAE Confirm message

has been verified, the *Rc* variable shall be set to the value of the Send-Confirm field, Sc shall be set to the value

216 – 1, the t1 (key expiration) timer shall be set, the t0 (retransmission) timer shall be canceled, send the Parent

process an Auth event, and the protocol instance shall transition to *Accepted* state.

*Fourteenth Erratum [Missing COM event]*

*1) Location:* §12.4.8.6.1, pg. 2471;

*2) Original Text: If one does, and it is in either Committed state or Confirmed state the frame shall be passed to the protocol instance.*

*3) Correction:* The sentence should be rewritten as follows: ’If one does, and it is in either Committed state or Confirmed state the frame shall be passed to the protocol instance, as COM event.’ with the final part being crucial to explain how it’s handled in the state machine.

*4) Rationale:* The protocol instance state machine produces transitions only if provided with an event. Without it, it would be impossible to begin parsing the frame passed by the Parent process. In the case of the Commit message, the protocol instance should receive a COM event.

And

*Fifteenth Erratum [Missing COM event]*

*1) Location:* §12.4.8.6.1, pg. 2471;

*2) Original Text: If an Anti-Clogging Token field exists and is correct, the parent process shall create a protocol instance.*

*3) Correction:* The sentence should be rewritten as follows: ’If an Anti-Clogging Token field exists and is correct, the parent process shall create a protocol instance and the frame shall be passed as a COM event.’ with the final part being crucial to explain how it’s handled in the state machine.

*4) Rationale:* As mentioned in the previous error, the state machine of a protocol instance always requires an event to accompany both the creation of the instance and the passing by the parent process of a message.

Discussion: Well, OK….

Proposed Resolution: Revised, instruct editor to modify §12.4.8.6.1 as indicated:

**12.4.8.6.1 Parent process behavior**

Upon receipt of an SAE Commit message, the parent process checks whether a protocol instance for the peer

MAC address exists in the database. If one does, and it is in either Committed state or Confirmed state the

frame shall be passed to the protocol instance as a Com event. If one does and it is in Accepted state, the scalar in the received frame is checked against the peer-scalar used in authentication of the existing protocol instance (in Accepted state). If it is identical, the frame shall be dropped. If not, the parent process checks the value of Open. If Open is greater than dot11RSNASAEAntiCloggingThreshold, the parent process shall check for the presence of an

Anti-Clogging Token field. If an Anti-Clogging Token field exists and is correct, the parent process shall create

a protocol instance and pass the frame to it as a Com event.

*Sixteenth Erratum [Missing Del event and state transition]*

*1) Location:* §12.4.8.6.3, pg. 2472;

*2) Original Text: If so and there is no password associated with that identifier, BadID shall be set and the protocol instance shall construct and transmit an Authentication frame with Status Code set to UNKNOWN\_PASSWORD\_IDENTIFIER.*

*3) Correction:* The sentence should be rewritten as follows: ’If so and there is no password associated with that identifier, BadID shall be set and the protocol instance shall construct and transmit an Authentication frame with Status Code set to UNKNOWN\_PASSWORD\_IDENTIFIER, and it shall send a Del event and remain in Nothing state.’

*4) Rationale:* This validation branch does not specify what actions the protocol instance should perform after sending the authentication frame. Leaving the interpretation open shows a shortcoming of the state machine thus conceived. The Nothing state takes on two meanings: the initial protocol state and the sink state pre-elimination of protocol instances. A DOS attack can be generated whenever a protocol instance enters or remains in the nothing state, without the event causing the transition, being a Del event. In this way, the attacker creates n instances of the same peer identity, violating the principle: ’For any given peer identity, there shall be only one protocol instance in Committed or Confirmed state.’ The best method for solving the problems introduced by incremental protocol updates would be to split the Nothing state semantics into two states: one that handles the creation of instances (Nothing state) and one that handles all cases in which instances have been terminated but not yet deleted (Removed state). Radical changes might be challenging to implement, so the proposed corrections will consider keeping the state machine in the current version.

Discussion: Yes, radical changes to the state machine are unwise and the comment does not warrant them. That said, an error is identified. It is best to make the behavior match other state machine behavior when errors are received in Nothing state—that is, send a Del event to the parent process and just leave everything else alone.

Proposed Resolution: Revised, instruct the editor to modify section §12.4.8.6.3 as indicated:

**12.4.8.6.3 Protocol instance behavior—Nothing state**

Upon receipt of a *Com* event, the protocol instance shall check the Status of the Authentication frame. If the Status code is not SUCCESS, the frame shall be silently discarded and a *Del* event shall be sent to the parent process. Otherwise, the frame shall be processed by first checking whether a password identifier is present. If so and there is no password associated with that identifier, *BadID* shall be set, the protocol instance shall construct and transmit an Authentication frame with Status Code set to UNKNOWN\_PASSWORD\_IDENTIFIER, and send a *Del* event to the parent process.

*Seventeenth Erratum [Missing status code or silent deletion]*

*1) Location:* §12.4.8.6.3, pg. 2472;

*2) Original Text: It shall then process the received SAE Commit message (see 12.4.5.4). If validation of the received SAE Commit message fails, the protocol instance shall send a Del event to the parent process;*

*3) Correction:* The sentence should be rewritten as follows: ’It shall then process the received SAE Commit message (see 12.4.5.4). If validation of the received SAE Commit message fails, the protocol instance shall send a Del event to the parent process and silently discard it;’

*4) Rationale:* Commit message handling can produce multiple errors: one due to the reception of unsupported groups, another due to the reception of scalars or elements not in the predetermined range and one caused by the reception of the identity element. Each error results in the rejection of the peer’s authentication. However, even though the wording recalls the definition of the Fail event, the event is not used for them. The event could be handled as, in this case, using a Del event, but what remains unspecified is whether or not the protocol instance should send an authentication frame containing the error status. In the literature, innumerable cases of poor handling of error messages led to attacks. This happens because implementations do not have precise guidance on how to handle the error and make choices that are sometimes incorrect, putting users’ security at risk. Therefore, to avoid possible vulnerability, all error cases must be handled. To handle this error, it must be decided whether to add new status codes or explicitly say that the message should be discarded silently.

Discussion: Not sure what other behavior could be imagined if a Del event is being sent to the parent process but saying to “silently discard [the frame]” can only help.

Proposed Resolution: Accept.

*Eighteenth Erratum [Algorithm identifier]*

*1) Location:* §12.4.8.6.4, pg. 2473;

*2) Original Text: If Sync is not greater than dot11RSNASAESync, Sync shall be incremented, an SAE Commit message with Status code equal to UNSUPPORTED FINITE CYCLIC GROUP indicating rejection, and the Algorithm identifier set to the rejected algorithm shall be sent to the peer, the t0 (retransmission) timer shall be set and the protocol instance shall remain in Committed state.*

*3) Correction:* Using a unique notation and clarifying which field or status code is the Authentication identifier in the authentication frame would be necessary.

*4) Rationale:* The ’algorithm identifier’ term was never used in the protocol specification, which makes it difficult to understand what it refers to.

Discussion: The rejected group is what intended to be indicated. Since the particular construction of the SAE Commit message depends on whether hash-to-element was used, it doesn’t make sense to repeat the if..else stuff here in the state machine, just note that the message indicates rejection and the constructed SAE Commit message will be correctly constructed.

Proposed Resolution: Revised, instruct the editor to modify §12.4.8.6.4 as indicated:

* If Sync is not greater than dot11RSNASAESync, Sync shall be incremented, an SAE Commit message indicating rejection of the offered group with Status code equal to UNSUPPORTED\_FINITE\_CYCLIC\_GROUP shall be sent to the peer, the t0 (retransmission) timer shall be set and the protocol instance shall remain in Committed state.

*Nineteenth Erratum [Missing Del event, status code or silent deletion]*

*1) Location:* §12.4.8.6.4, pg. 2473;

*2) Original Text: The mesh STA, with the numerically lesser of the two MAC addresses, zeros Sync, shall increment Sc, choose the group from the received SAE Commit message, generate new PWE and new secret values according to 12.4.5.2, process the received SAE Commit message according to 12.4.5.4, generate a new SAE Commit message and SAE Confirm message, and shall transmit the new Commit and Confirm to the peer. It shall then transition to Confirmed.*

and

*If the received element and scalar differ from the element and scalar offered, the received SAE Commit message shall be processed according to 12.4.5.4, the Sc counter shall be incremented (thereby setting its value to one), the protocol instance shall then construct an SAE Confirm message, transmit.*

*3) Correction:* After the following sentence: ’SAE Commit message according to 12.4.5.4’ should be added: ’the protocol instance shall send a Del event to the parent process and silently discard the message;’

*4) Rationale:* As specified in the previous errata, it is necessary to handle all the errors that may arise; otherwise, unexpected behaviour in the state machine may occur. In these two cases, without a Del event, an implementation could mistakenly choose to continue the execution and transition to the next state or stop without providing any information about its current state to the parent process.

Discussion: the “differ from the element and scalar offered” is to prevent against reflection attacks. The element and scalar in the SAE Commit messages from both sides cannot be equal. So we don’t want to discard the message and send a Del event.

Proposed Resolution: Reject.

*Twentieth Erratum [Rejection frames]*

*1) Location:* §12.4.8.6.4, pg. 2473;

*2) Original Text: Rejection frames received in Confirmed state shall be silently discarded.*

*3) Correction:* Replace the term ’Rejection frames’ with another wording already used in the specification.

*4) Rationale:* The term ’Rejection frames’ has never been used elsewhere in the specification. It would be appropriate to use a term already present in the specification (e.g. Authentication frame with status code different from SUCCESS) to make the specification more understandable.

Discussion: This condition should be handled by the next paragraph and this can be safely removed. But this comment illustrates another problem: we are handling a status of SAE\_HASH\_TO\_ELEMENT as an error when it’s not. Need to fix that.

Proposed Resolution: Instruct the editor to modify sections §12.4.8.6.4 and §12.4.8.6.5 as indicated:

**12.4.8.6.4 Protocol instance behavior—Committed state**

* If the Status is a nonzero value other than SAE\_HASH\_TO\_ELEMENT, the frame shall be silently discarded and the t0 (retransmission) timer shall be set.
* If the Status is zero or SAE\_HASH\_TO\_ELEMENT, the finite cyclic group field is checked. If the group is not supported, BadGrp shall be set and the value of Sync shall be checked.

**12.4.8.6.5 Protocol instance behavior—Confirmed state**

Upon receipt of a Com event, the t0 (retransmission) timer shall be canceled. If the Status is a nonzero value other than SAE\_HASH\_TO\_ELEMENT, the frame shall be silently discarded, the t0 (retransmission) timer set, and the protocol instance shall remain in the Confirmed state.

**References:**