IEEE P802.11  
Wireless LANs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Resolution for CIDs related to Protected BA Procedure | | | | |
| Date: January 14, 2021 | | | | |
| Author(s): | | | | |
| Name | Affiliation | Address | Phone | email |
| Abhishek Patil | Qualcomm Inc |  |  | appatil@qti.qualcomm.com |
| Alfred Asterjadhi |  |  |  |
| George Cherian |  |  |  |
| Jouni Malinen |  |  |  |
| Duncan Ho |  |  |  |
| Gaurang Naik |  |  |  |
| Yanjun Sun |  |  |  |
| Abdel Karim |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

Abstract

This submission proposes resolutions for comments received in LB258 (for REVme D1.0) related to protected BA feature:

***TGm editor: The baseline for this document is REVme D1.0.***

**Revisions:**

* Rev 0: Initial version of the document.

Interpretation of a Motion to Adopt

A motion to approve this submission means that the editing instructions and any changed or added material are actioned in the TGm Draft. This introduction is not part of the adopted material.

Part 1

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **CID** | **Commenter** | **Page** | **Line** | **Clause** | **Comment** | **Proposed Change** | **Resolution** |
| 1002 | Abhishek Patil | 1300.00 | 64 | 9.4.2.24.4 | Based on discussion with several members (affiliated with different organizations), it was discovered that many devices (already deployed in the field) are incorrectly setting the PBAC bit in RSN Capabilities field to 1 even when they do not support protected BA operation. Future amendments are expects to mandate support for protected BA procedure (given the increasing vulnerabilities). Incorrect setting of PBAC bit will create inter-op issues when the peer device that supports protected BA procedure, expects the other side to follow the same procedure. | The commenter will provide a contribution to address this issue. | **Revised**  Agree with the comment. Advertising PBAC while the device doesn’t support the procedure for protected BA agreement will lead to interop issues when the peer device supports this feature. Future amendments are expected to encourage protected BA agreements to thwart any attacks that exploit the vulnerabilities of BAR and existing BA procedure.  The proposed change sets the exiting PBAC bit to reserve and defines a new bit in the RSN Extension element.  **TGm editor, please implement changes as shown in doc 11-22/0082r0 tagged as 1002** |
| 1821 | Mark RISON | 2282.00 | 60 | 10.25.7 | "A STA indicates support for protected block ack by setting the RSN Capabilities field subfields MFPC, MFPR and PBAC to 1. " -- it is not clear why MFPR need be 1 (at least if dot11RSNAPBACRequired is not true) | Delete ", MFPR" from the cited text | **Revised**  Agree with the comment. A PBAC STA can choose to negotiate a BA agreement w/ a non-PBAC STA – see REVme D1.0 P2283L5. Therefore, the requirement of MFPR set 1 does not apply to a PBAC STA  **TGm editor, please implement changes as shown in doc 11-22/0082r0 tagged as 1821** |

**Discussion:**

The IEEE 802.11 standard defines a mechanism to setup protected block ack (BA) session between an originator and a recipient. A device indicates support for protected BA by setting the PBAC, MFPR and MFPC subfield in the RSN Capabilities field. However, it was determined that there exist devices in the field which can incorrectly indicate support for protected BA, by setting the related bits if the peer STA has set them to 1. This came to light during debugging of some other issues related to RSN capabilities field and when additions tests were run where the peer device was configured to indicate support for protected BA. Such ‘copying’ of PBAC capability will lead to inter-op issues when a peer device genuinely supports protected BA feature. Fortunately, we are not aware of any existing implementations of protected BA. This is based on our discussion (for several months) with various members, affiliated with different 802.11 product vendors. In addition, an email was sent on January 13th, 2022, to the IEEE 802.11 reflector soliciting a response if anyone was aware of an existing implementation of protected BA. We have not received any responses until the time this document was updated (January 17th, 2022). As a result, we don’t expect any impact to existing devices, in the field, with the signaling changes proposed in this document.

* RSN capabilities[1002]

***TGm editor: Please update Figure 9-350 to change bit 12 (B12) from “PBAC” to “Reserved”***

***TGm editor: Please update the following bullet in this subclause as shown below:***

* Bit 12: Reserved.
* RSN Extension element (RSNXE)[1002]

***TGm editor: Please add a new row to Table 9-363 as shown below and accordingly update the content of the last row corresponding to ‘Reserved’ values:***

|  |  |  |
| --- | --- | --- |
| * Extended RSN Capabilities field | | |
| Bit | Information | Notes |
| <ANA> | PBAC | A STA sets the PBAC field to 1 to indicate it can establish a protected block ack agreement and sets it to 0 otherwise. |

* **Protected block ack agreement**

***TGm editor: Please update the 1st paragraph this subclause as shown below:***

[1002, 1821]A STA indicates support for protected block ack by setting the MFPC subfield in the RSN Capabilities field to 1 (see 9.4.2.24.4 (RSN Capabilities)) and the PBAC subfield in Extended RSN Capabilities field to 1 (see 9.4.2.241 (RSN Extension element (RSNXE))). Such a STA is a PBAC STA; otherwise, the STA is a non-PBAC STA. A block ack agreement that is successfully negotiated between two PBAC STAs is a protected block ack agreement. A block ack agreement that is successfully negotiated between two STAs when either or both of the STAs is not a PBAC STA is a block ack agreement that is not a protected block ack agreement.

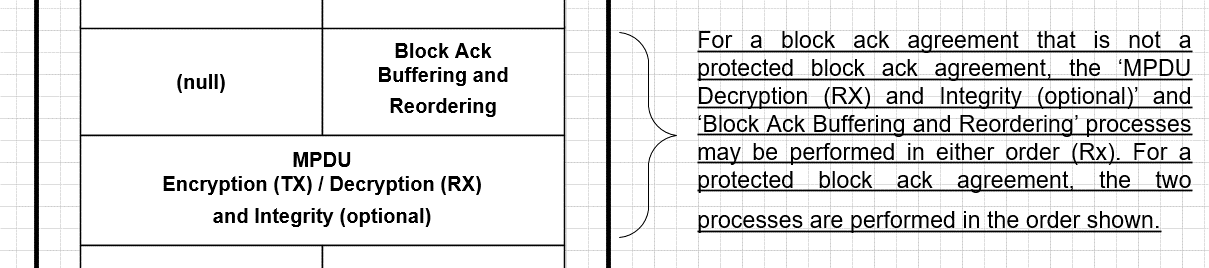
Part 2

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **CID** | **Commenter** | **Page** | **Line** | **Clause** | **Comment** | **Proposed Change** | **Resolution** |
| 1001 | Abhishek Patil | 355.00 | 45 | 5.1.5.1 | In a protected BA agreement, the order of decryption and update to reorder buffer must be maintained (as shown in Figure 5-1). This will prevent an attack scenario where a rogue device injects fake Data frame(s) which causes WinStartB to get updated while the decryption of the frame(s) fails. Update the side note shown in the figure to capture this intention. | The commenter will provide a contribution to address this. | **Revised**  Agree with the comment. Changing the order of decryption/integrity check and reorder buffering will enable an attack scenario where the reorder buffer is filled with (fake) Data frames and the attacker is able to update the WinStartB value.  The proposed change updates the side note in Figure 5-1 requiring that the order of the processes is maintained in protected BA agreement.  **TGm editor, please implement changes as shown in doc 11-22/0082r0 tagged as 1001** |

**5.1.5 MAC data service architecture**

**5.1.5.1 General**[1001]

***TGm editor: Please replace the side note in Figure 5-1 as shown below:***



Part 3

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **CID** | **Commenter** | **Page** | **Line** | **Clause** | **Comment** | **Proposed Change** | **Resolution** |
| 1014 | Abhishek Patil | 2283.00 | 26 | 10.25.7 | A BAR frame is not protected. Therefore, in a protected BA setup, the Block Ack Starting Sequence Control subfield value in a BAR frame must not be used for updating the WinStartR value. | The commenter will provide a contribution to address this. | **Revised**  Agree with the comment. Since BAR is not a protected frame, the proposed change states that the recipient STA must not use the value carried in Block Ack Starting Sequence Control subfield to update the value of WinStartR. Instead, the WinStartR is updated based on the SN carried in a genuine MPDU (see resolutions to CIDs 1015 and 1016).  **TGm editor, please implement changes as shown in doc 11-22/0082r0 tagged as 1014** |
| 1745 | Mark RISON | 2283.00 | 24 | 10.25.7 | "The Block Ack Starting Sequence Control subfield value may be utilized for the purposes of updating the value of WinStartR." is not clear. This is generally the case anyway | Prepend "NOTE---" and change "may" to "could" | **Revised**  The cited sentence was updated as a resolution to CID 1014. Since BAR is not a protected frame, the proposed change states that the recipient STA must not use the value carried in Block Ack Starting Sequence Control subfield to update the value of WinStartR.  **TGm editor, please implement changes as shown in doc 11-22/0082r0 tagged as 1014** |
| 1017 | Abhishek Patil | 2283.00 | 31 | 10.25.7 | WinStartB must not be updated when an error/attack condition is determined (i.e., dot11PBACErrors is incremented). | The commenter will provide a contribution to address this. | **Revised**  Agree with the comment. The proposed change states that WinStartB is not updated when dot11PBACErrors is incremented.  **TGm editor, please implement changes as shown in doc 11-22/0082r0 tagged as 1017** |
| 1015 | Abhishek Patil | 2283.00 | 31 | 10.25.7 | In addition to the stated conditions (P2283L22), the scoreboard context at the recipient STA must not be updated if MPDU decryption or integrity check fails. This can happen when an attacker injects a fake Data frame in such case, the decryption will fail). | The commenter will provide a contribution to address this. | **Revised**  Agree with the comment. The proposed change states that the recipient STA flushes the entire scoreboard context if it maintains partial state and the MPDU decryption or integrity check fails, or the recipient STA flushes the entry for the MPDU from the scoreboard context if it maintains full state and the decryption or integrity check for that MPDU fails.  **TGm editor, please implement changes as shown in doc 11-22/0082r0 tagged as 1015** |
| 1016 | Abhishek Patil | 2283.00 | 31 | 10.25.7 | In addition to the stated conditions (P2283L22), the scoreboard context and the reorder buffer at the recipient STA must not be updated if MPDU fails replay check. This can happen when an attacker replays a valid MPDU with an updated SN. Such MPDU will pass decryption but will fail replay check. | The commenter will provide a contribution to address this. | **Revised**  Agree with the comment. The proposed change states that the recipient STA flushes the entire scoreboard context and flush the entry for the MPDU from reorder buffer if it maintains partial state and the PN-based replay check failed for an MPDU that passed decryption and integrity check, or the recipient STA flushes the entry for the MPDU from the scoreboard context and the reorder buffer if it maintains full state and the PN-based replay check fails for an MPDU that passed decryption and integrity check.  **TGm editor, please implement changes as shown in doc 11-22/0082r0 tagged as 1016** |

* **Protected block ack agreement**

***TGm editor: Please update the 4th paragraph this subclause as shown below:***

A STA that has successfully negotiated a protected block ack agreement, for a certain TID, shall obey the following rules for that agreement as a block ack recipient in addition to rules specified in 10.25.6.3 (Scoreboard context control during full-state operation) to 10.25.6.6 (Receive reordering buffer control operation):

* The STA shall respond to a BlockAckReq frame from the originator with an immediate BlockAck frame. The Block Ack Starting Sequence Control subfield value shall be ignored for the purposes of updating the value of *WinStartB* and[1014] *WinStartR*. If the Block Ack Starting Sequence Control subfield value is greater than *WinEndB* or less than *WinStartB*, dot11PBACErrors shall be incremented by 1 and *WinStartB* shall not be updated[1017]. If, for a block ack agreement with segmentation and reassembly, the MPDU Starting Sequence subfield value is greater than *WinEndB* or less than *WinStartB*, dot11PBACErrors shall be incremented by 1 and *WinStartB* shall not be updated[1017].
* [1015, 1016]The STA shall respond to an MPDU from the originator, that has an ack policy other than No Ack or Block Ack or HETP Ack, with a BlockAck frame. If the STA maintains full state operation and an MPDU does not pass decryption or integrity check, then the STA shall not update the value of *WinStartR*, shall clear the scoreboard context for that MPDU, and shall increment dot11PBACErrors by 1. If the STA maintains partial state operation and an MPDU does not pass decryption or integrity check, then the STA shall clear the entire scoreboard context, and shall increment dot11PBACErrors by 1. If the STA maintains full state operation and the replay check fails for an MPDU that was successfully decrypted and passed integrity check, then the STA shall not update the value of *WinStartB* and *WinStartR*, shall clear the scoreboard context for that MPDU, shall clear the entry for that MPDU from the reorder buffer, and shall increment dot11PBACErrors by 1. If the STA maintains partial state operation and the replay check fails for an MPDU that was successfully decrypted and passed integrity check, then the STA shall clear the entire scoreboard context, shall not update the value of *WinStartB*, shall clear the entry for that MPDU from the reorder buffer, and shall increment dot11PBACErrors by 1.
* Upon receipt of a valid robust ADDBA Request frame for an established protected block ack agreement whose TID and transmitter address are the same as those of the block ack agreement, the STA shall update its *WinStartR* and *WinStartB* values based on the starting sequence number in the robust ADDBA Request frame according to the procedures outlined for reception of BlockAckReq frames in 10.25.6.3 (Scoreboard context control during full-state operation), 10.25.6.4 (Scoreboard context control during partial-state operation), 10.25.6.6.1 (General), and 10.25.6.6.3 (Operation for each received BlockAckReq), while treating the starting sequence number as though it were the *SSN* of a received BlockAckReq frame or, in case of a block ack agreement with segmentation and reassembly, treating the MPDU starting sequence number as though it were the MPDU SSN of a received BlockAckReq frame. Values in other fields of the ADDBA Request frame shall be ignored.

Part 4

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **CID** | **Commenter** | **Page** | **Line** | **Clause** | **Comment** | **Proposed Change** | **Resolution** |
| 1013 | Abhishek Patil | 2283.00 | 15 | 10.25.7 | At a receiver, Management frames (such as ADDBA Request) are handled by processes that come much later in the sequence while an ACK response is sent in SIFS. To prevent race condition where the originator and recipient get into different states, the originator must not update the WinStartO until it receives a confirmation (ADDBA Response frame) from the recipient in response to its ADDBA Request frame sent to update the WinStartB. Clause 9.6.4.1 (P1860L51) mentions this but the normative text and details of the procedure are not described in clause 10.25.7. | The commenter will provide a contribution to address this issue. | **Under discussion!** |

* **Protected block ack agreement**

***TGm editor: Please update the 3rd paragraph this subclause as shown below:***

A STA that has successfully negotiated a protected block ack agreement, for a certain TID, shall obey the following rules for that agreement as a block ack originator in addition to rules specified in 10.25.6.7 (Originator’s behavior) and 10.25.6.8 (Maintaining block ack state at the originator):

* To change the value of *WinStartB* at the receiver, the STA shall use a ADDBA Request frame
* Upon reception of a robust ADDBA Response frame, in response to its robust ADDBA Request frame, for an established protected block ack agreement whose TID and transmitter address are the same as those of the block ack agreement, the STA shall validate the value carried in the Dialog Token field of the ADDBA Response frame before updating its *WinStartO* with the value of the Block Ack Starting Sequence Control field of the ADDBA Request frame it transmitted. The STA shall ignore the values carried in other fields of the Action field of the ADDBA Response frame except for the Block Ack Timeout Value field.

NOTE – An originator STA can set the Block Ack Timeout Value field in an ADDBA Request frame to a value that is different from the most recently negotiated one if it wants to update the block ack timeout value (see 10.25.2 (Setup and modification of the block ack parameters)). Otherwise, the STA sets the field to the same value as the negotiated block ack timeout.

***TGm editor: Please update the 3rd bullet of the 4th paragraph this subclause as shown below:***

* Upon receipt of a ADDBA Request frame for an established protected block ack agreement whose TID and transmitter address are the same as those of the block ack agreement, the STA shall update its *WinStartR* and *WinStartB* values based on the starting sequence number in the robust ADDBA Request frame according to the procedures outlined for reception of BlockAckReq frames in 10.25.6.3 (Scoreboard context control during full-state operation), 10.25.6.4 (Scoreboard context control during partial-state operation), 10.25.6.6.1 (General), and 10.25.6.6.3 (Operation for each received BlockAckReq), while treating the starting sequence number as though it were the *SSN* of a received BlockAckReq frame or, in case of a block ack agreement with segmentation and reassembly, treating the MPDU starting sequence number as though it were the MPDU SSN of a received BlockAckReq frame and shall transmit a robust ADDBA Response frame. The STA shall ignore the values carried in the other fields of the Action field of the ADDBA Request frame, except for the Block Ack Timeout Value field (also see 10.25.2 (Setup and modification of the block ack parameters)).