IEEE P802.11
Wireless LANs

|  |
| --- |
| 11bi D2.0 CR for CIDs related to PGTK |
| Date: 2025-10-15 |
| Author(s): |
| Name | Affiliation | Address | Phone | Email |
| Julien Sevin | Canon  |  |  | julien.sevin@crf.canon.fr |
| Stéphane Baron |  |  | stephane.baron@crf.canon.fr |
| Patrice Nezou |  |  | patrice.nezou@crf.canon.fr |

Abstract

This submission proposes resolutions for the following CIDs:

2334, 2348, 2487, 2490, 2491

Revisions:

* Rev 0. Initial version of the document.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **CID** | **Clause** | **Comment** | **Proposed Change** | **Resolution** |
| 2334 | 12.2.4 | What does it mean to "program a key"? The baseline terminology is to "install a key" | Make the following changes:At 129.48, 129.50, 129.58, 129.60, 130.2, and 130.4" change "programs" to "installs"At 129.52 and 129.61, change "programmed into" to "installed in" | ACCEPTEDRemark : the term "program a key" has been initially used to be in line with the terms already used in clause 12.2.4 However, only the clause 12.2.4. uses this term in the baseline. A specific CID [#233] in the LB289 - IEEE P802.11REVmf D1.0 Initial WG Letter Ballot has been done to fix the use of the term “programs” in the baselineInstructions to the editor: Please make the changes as shown under CID 2334 in this document |
| 2487 | 3.2 | PGTK definition says that it is used for frame anonymization, but does not specify which type of FA , for CPE of BPE."[PGTK] A random value, assigned by an access point (AP) multilink device (MLD), shared to all non-access point (non-AP) multi-link devices (MLDs) associated to the APMLD for frame anonymization."In 10.71, PGTK is only used for BPE\_MHA\_block and not used for CPE\_MHA\_block. |  Clarify in PGTK definition whether it is used for both CPE and BPE FA. | REJECTEDThe term frame anonymization is used in the standard to encompass both CPE frame anonymization and BPE frame anonymization and the PGTK is used for both.  |
| 2491 | 6.5.14.1.4 | "In the text ""When the Key Type parameteris PGTK, the MAC installs the key, and the FA is processed using that key"", what does it mean by processing FA with PGTK? Does it mean CPE\_MHA\_block or BPE\_MHA\_block generated using PGTK? Could the text be more specific in how PGTK is used for FA, perhaps by adding a NOTE." | As in comment | REJECTEDNo change in order to stay aligned with the definition of the PGTK and the other parts of the standard which indicates only that the PGTK is used for FA in order to encompass its use both in the EPP Epoch Start Time Computation and the BPE parameters computation. |
| 2490 | 12.7.2 | it may provide more flexibility to define PGTK as variable size instead of fixed 32 octets. Since PGTK KDE is encapsulated in a KDE (FIG 12-38 in REVmf) there is Length field that would indicate size of PGTK. This would be similar to GTK or IGTK both of which have variable size. | Change PGTK size from 32 octets to variable. | ACCEPTEDInstructions to the editor: Please make the changes as shown under CID 2490 in this document. |
| 2348 | 10.71.4 | PGTK is used for two privacy purposes (EPP Epoch Start Time Computation and Establishing BPE MAC header anonymization parameter sets) which constitutes a bad key hygiene / key separation in terms of security. | Please define two different keys derived from PGTK, one for the EPP Epoch Start Time Computation and one for Establishing BPE MAC header anonymization parameter sets | REVISEDIntroduction of two different keys, both derived from PGTK , one (ST\_PGDK) for the EPP Epoch Start Time Computation and one (BPE\_PGDK) for Establishing BPE MAC header anonymization parameter setsInstructions to the editor: Please make the changes as shown under CID 2348 in this document. |

* **EPP Epoch Start Time Computation**

***TGbi Editor: Modify the sixth paragraph of the section as followed :***

EpochTSFStartTime(*n*) = PlannedEpochTSFStartTime(*n*) for the link + ΔIT(*n*)

where

PlannedEpochTSFStartTime(*n*) = FirstPlannedEpochTSFStartTime + (*n* – EpochNumberOffset) × EpochInterval mod 264

ΔIT(*n*) = int (KDF-*Hash*-*Length*(ST\_PGDK (#2348), “ERCM”, Seed + (*n* × EpochInterval))) mod TimeRangeTU

and where

*n* is the current number of the EPP epoch in the EPP epoch sequence.

PlannedEpochTSFStartTime(*n*) is the TSF timer value of the link corresponding to the nominal start

 time of the EPP epoch number n in the EPP epoch sequence.

 This planned start time occurs at a regular time interval equal to the

 epoch interval.

EpochNumberOffset is the value indicated in the Epoch Number Offset field of the

 EPP Epoch Settings field.

EpochInterval is the value in TU corresponding to the Epoch Interval

 field of the EPP Epoch Settings field .

KDF-*Hash*-*Length* is the key derivation function as defined in

 12.7.1.6.2 (Key derivation function (KDF)) using the

 hash algorithm identified by the AKM suite selector

 (see -- Editor Note: (ANA assignment is done)).

*Length* is the number of bits to derive. 16 bits are derived for ΔIT.

FirstPlannedEpochTSFStartTime is the value of the first epoch TSF start time,

 initialized, upon reception of an EPP element by the STA with

 the First Epoch TSF Start Time value of the EPP element of

 the received EPP Epoch Settings field.

TimeRangeTU is the value in TU corresponding to the value of the Epoch Start Time

 Variation Range field multiplied by the number of TU in the Epoch

 Interval Unit field of the EPP Epoch Settings field.

ST\_PGDK is the Privacy Group Derivation Key used for Key EPP Epoch Start Time Computation, derived from PGTK as defined in 12.7.1.9 (Privacy group key hierarchy). (#2348)

Seed is the Group Epoch Seed field value of the received EPP Epoch Settings

 field.

* **Establishing BPE MAC header anonymization parameter sets**

***TGbi Editor: Modify the third paragraph of the section as followed :***

For a given EPP epoch, the BPE\_MHA\_block shall be generated as:

BPE\_MHA\_block = KDF-*Hash*-*Length* (BPE\_PGDK (#2348), “BPE\_MHA\_block”, Seed + (*n* × EpochInterval)),

where

KDF-*Hash*-*Length* is the key derivation function as defined in 12.7.1.6.2 (Key derivation

 function (KDF)) using the hash algorithm identified by the AKM suite

 selector (see Table 9-190 (AKM suite selectors))

BPE\_PGDK is the Privacy Group Derivation Key used for Establishing BPE MAC header anonymization parameter sets, derived from PGTK as defined in 12.7.1.9 (Privacy group key hierarchy) (#2348)

*n* is the current number of the EPP epoch in the EPP epoch sequence as

 defined in 10.71.2.4 (EPP Epoch Start Time Computation)

*Length* is the total number of bits to derive. A total of 960 bits are derived for a

 BPE\_MHA\_block.

Seed is the value of the Group Epoch Seed field of the received EPP Epoch Settings

 field.

* **Security**
* **Framework**
* **RSNA establishment**

***Change the first bullet, the second bullet, the fifth bullet, and add a new bullet of the first paragraph as follows (not all lines are shown):***

An SME establishes an RSNA in one of seven ways:

* If an RSNA uses authentication negotiated over IEEE Std 802.1X or FILS authentication in an infrastructure BSS, an SME establishes an RSNA as follows:
* It identifies the AP as an RSNA AP from the AP’s Beacon, Privacy Beacon, DMG Beacon, Announce, Information Response, FILS Discovery, or Probe Response frames.

NOTE—As described in 10.71.8.4 (Steering to BPE AP MLD), the STA might identify an AP affiliated with a BPE AP MLD by using the AP's anonymized link address and the Neighbor Report element that is received from the associated AP.

* It shall invoke Open System, IEEE 802.1X authentication, or FILS authentication if the STA is a non-DMG STA.
* It negotiates cipher suites during the association process, as described in 12.6.2 (RSNA selection) and 12.6.3 (RSNA policy selection in an infrastructure BSS).
* It uses IEEE Std 802.1X-2020 to authenticate if IEEE 802.1X authentication is not performed before association, as described in 12.6.8 (RSNA establishment in an infrastructure BSS) and 12.6.9 (RSNA authentication in an IBSS), FT protocol to authenticate as described in 13.5 (FT protocol) or uses FILS authentication to authenticate as described in 12.11 (Authentication for FILS).
* If EPP epoch operation is supported by both the AP MLD and the non-AP MLD, the SME installs(#2334) the PGTK into the MAC for frame anonymization.
* If the AP is affiliated with a BPE AP MLD, the SME installs(#2334) the identity key if the key is not already installed in(#2334) the MAC to identify the AP MLDs with that configured identity key from its Privacy Beacon frames.
* If an RSNA is based on a PSK or password in an infrastructure BSS, an SME establishes an RSNA as follows:
* If EPP epoch operation is supported by both the AP MLD and the non-AP MLD, the SME installs(#2334) the PGTK into the MAC for frame anonymization.
* If the AP is affiliated with a BPE AP MLD, the SME installs(#2334)the identity key if the key is not already installed in(#2334) the MAC for identifying the AP MLDs with that configured identity key from its Privacy Beacon frames.
* If an RSNA allows for confidentiality only (no authentication) in an infrastructure BSS, an SME establishes an RSNA as follows:
* If EPP epoch operation is supported by both the AP MLD and the non-AP MLD, the SME installs(#2334) the PGTK into the MAC for frame anonymization.
* If the AP is affiliated with a BPE AP MLD, the SME installs(#2334) the identity key into the MAC for identifying the AP MLD from its Privacy Beacon frames.
* If an RSNA uses PASN authentication, an RSNA capable the STA establishes an RSNA as described in 12.13 (Preassociation security negotiation(11az)).
* If an RSNA uses EPPKE authentication, an RSNA capable STA establishes an RSNA as described in 12.16.9 (Enhanced Privacy Protection Key Exchange).
* Keys and key distribution
* Key hierarchy

12.7.1.9 Privacy group key hierarchy

***TGbi Editor: Insert the following paragraph at the end of the section***

A ST\_PGDK and a BPE\_PGDK are derived from the PGTK using the KDF defined in 12.7.1.6.2 (Key derivation function (KDF))):

ST\_PGDK || BPE\_PGDK = KDF-Hash-Length(PGTK, “PGTK”)

where

— KDF-Hash-Length is the key derivation function as defined in 12.7.1.6.2 (Key derivation function (KDF)) using the hash algorithm identified by the AKM suite selector (see Table 9-190 (AKM suite selectors)).

— The "Length" of the KDF is equal to twice the length of the PGTK.

ST\_PGDK is used for EPP Epoch Start Time Computation as specified in section 10.71.2.3 (EPP epoch transition operations) and BPE\_PGDK is used for BPE frame anonymization as specified in 10.71.4 (Establishing BPE MAC header anonymization parameter sets). (#2348)

* **EAPOL-Key frames**

***TGbi Editor: Modify the following paragraph at the end of 12.7.2 (EAPOL-Key frames)***

The format of the PGTK KDE is shown in Figure 12-50i (PGTK KDE format).

|  |  |  |
| --- | --- | --- |
|  | PGTK Switch Time  | PGTK |
| Octets: | 8 | variable(#2490)  |
| * **PGTK KDE format**
 |