IEEE P802.11
Wireless LANs

|  |
| --- |
| 11bh D0.2 CR for STA generating ID and pre-association identification |
| Date: 2022-12-13 |
| Author(s): |
| Name | Affiliation | Address | Phone | email |
| Jay Yang | Nokia |  |  | yang.zhijie@nokia-sbell.com |
| Orhan Okan Mutgan | Nokia |  |  | okan.mutgan@nokia-sbell.com |
| Liu Jianguo | Nokia |  |  | jianguo.a.liu@nokia-sbell.com |

Abstract

This submission proposes resolutions for the following CIDs:

7, 9, 36, 40, 41, 42, 61, 64,

Revisions:

* Rev 0: Initial version of the document.
* Rev 1: add CID 7 and 11bh draft0.2 context in the proposed change
* Rev2: add the new SP results in the discussion part.
* Rev3: remove CID 19 and 20 to a separate contribution, remove 65 as it’s addressed in 1329r2.
* Rev4: address some editorial issue.
* Rev5: add e-RRCM solution based on the comments from some members in last interim meeting.
* Rev6: revised on e-RRCM solution based on 1982r1.
* Rev7: Editorial change in the first page (latest date, typo fixed: 1982r1**1802r1**, e-RRCM proposed text  is aligned with the ideas in **1802r1**)

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 TGbh D0.2 Draft. This introduction is not part of the adopted material.

***Editing instructions formatted like this are intended to be copied into the TGbh D0.2 Draft. (i.e. they are instructions to the 802.11 editor on how to merge the text with the baseline documents).***

***TGbh Editor: Editing instructions preceded by “TGbh Editor” are instructions to the TGbh editor to modify existing material in the TGbh draft. As a result of adopting the changes, the TGbh editor will execute the instructions rather than copy them to the TGbh Draft.***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **CID** | **Commenter** | **Comment** | **Proposed Change** | **Resolution** |
| 7 | Jay Yang | Device ID only can be used in post association, we need to a solution to cover the probing case. | the commenter will provide a solution. | Revised—Agree in principle with the commenter. The identifiable probe is widely used like use case 4.1,case 4.8 and case 4.26 in the approved use cases tracking document 332R37.Some member wants more discussion on use case 4.1. Here we provide the resolution to cover the identifiable probe in post-association to meet the requirement of use case 4.8 and 4.26, which are in the scope of 11bh group.TGbh editor to make the changes shown in 11-22/1079r7  |
| 9 | Jay Yang | an AP may not grant an identifier to some STAs once it's recognized via the MAC address.RMA causes such implement broken, need to provide a solution to address it. | the commenter will provide a solution on it. | Revised – Agree in principle with the commenter. There is no need extra identifier carried in the eapol frame if the STA use the identified RMA.11bh SPEC should provide an STA generated RMAs solution to meet such requirementTGbh editor to make the changes shown in 11-22/1079r7  |
|  |  |  |  |  |
|  |  |  |  |  |
| 36 | Julien Sevin | No privacy enhancement mechanism is specified for covering the pre-association use cases specified by the 11bh group in the document 332r37. | Specify a privacy enhancement mechanism for identifying a STA operating with a Random MAC Address before the association | Revised – Agree in principle with the commenter. 11bh SPEC should provide an STA generated identifier solution to cover the pre- association case .TGbh editor to make the changes shown in 11-22/1079r7  |
| 40 | stephane baron | Issue tracking document contains scenario (especially 4.2 : returning device) agreed by the group and that is not addressed by the current draft. | Provide a mechanism that supports scenario 4.2 by providing a MAC address based mechanism that allows a returning station to be recognized. | Revised – Agree in principle with the commenter. Access Control during the associating requires the authentication/(re)association frame request frame carrying the identified RMA.11bh SPEC should provide an STA generated identifier solution to cover the pre- association case .TGbh editor to make the changes shown in 11-22/1079r7  |
| 41 | Patrice Nezou | The document 332r37 describes some scenarios related to the pre-association procedure. The current draft does not propose any privacy enhancement during this procedure. | Need additional mechanisms to enhance the privacy during the pre-association procedure. | Revised – Agree in principle with the commenter. 11bh SPEC should provide an STA generated identifier solution to cover the pre- association case .TGbh editor to make the changes shown in 11-22/1079r7  |
| 42 | Mikael Lorgeoux | A mechanism for privacy enhancement is missing for the coverage of pre-association use cases specified in the doc 332r37. | Specify a privacy enhancement mechanism for covering pre-association use cases of doc 332r37. | Revised – Agree in principle with the commenter. 11bh SPEC should provide an STA generated identifier solution to cover the pre- association case .TGbh editor to make the changes shown in 11-22/1079r7  |
| 61 | Mark Hamilton | With majority support for a STA-generated device ID (for example, Motion #3, although not 75% on a particular proposal, yet) and evidence that both network-generated and STA-generated can coexist (cf 11-22/0908), add a STA generated ID variant. | Add a STA-generated Device ID variant, and appropriate mechanism (if needed) to select an operating mode. | Agree in principle with the commenter. 11bh SPEC should provide a STA controlled device ID variant solution.TGbh editor to make the changes shown in 11-22/1079r7  |
| 64 | Jarkko Kneckt | The 802.11bh should define a protocol that allows STA to provide STA ID that the STA uses to identify itself to the AP in the following authentications/associations. | Please allow STA to have a possibility to provide to AP the STA Identifier that is used to identify the STA. | Agree in principle with the commenter. 11bh SPEC should provide a STA controlled device ID variant solution.Proposed resolution is in line with the proposed change.TGbh editor to make the changes shown in 11-22/1079r7  |
| ~~65~~ | ~~Jarkko Kneckt~~ | ~~The STA Identifier should be taken into use only if the STA opts-in to use the identifier. Currently AP may just push a STA ID for the STA even if the STA does not want to have it.~~ | ~~Please allow the STA to have control on whether it desires to use STA identifier in the following authentications and associations.~~ |  |

**Discussion:**

**SP in the plenary meeting in Bangkok:**

**Do you support TGbh working on adding an additional mechanism to support identification prior to association (or at Association Request)?**

**Y: 17 N: 4 Abs: 7**

**However, we still can have MAAD scheme(network controlled solution) to cover the pre-association use cases. This PDT provide three options:**

**Opt1: RRCM only**

**Opt2: e-RRCM only**

**Opt3: RRCM+MAAD.**

**The merit of e-RRCM**

1)the generated RMA is used as index to map the pairwise key, RMA is never the identifier.

2)address trackable issue based on the RMAs

3)Mutual identification following the recommended direction in DPP.

4) void DOS (Denial of Service) attack.

5) fast search the pairwise key, MIC only calculate on time.

6) address the identifiter copy issue, replay attack, spoof AP issue.

# Proposed text change(Opt1:RRCM)

***1) Add following definition to 3.2***

**Rule-based Random and Changing MAC Address (RRCM):** A privacy enhancement mechanism for non-AP STA and AP to generate one or more Random Mac Addresses (RMA) for use by non-AP STA to prevent non-AP STA from being tracked (by third parties) and still allow the non-AP STA to be identified by the AP in subsequent message exchanges. “Rule-based” implies that the non-AP STA and AP apply the same procedures for generating RMA or RMA(s) locally at their sides.

**RMAK (RMA Key):** RMAK is the key that is used to generate one or more Random Mac Addresses (RMA) for RRCM procedure.

***2) Add a new capability information to Table 9-363 Extended Capabilities field***

|  |  |  |
| --- | --- | --- |
| **Bit** | **Information** | **Notes** |
| <ANA> | Device ID Support | The STA sets the Device ID Support field to 1 to indicate support for Device ID indication. Otherwise, the STA sets the Device ID field to 0. |
| <ANA> | RRCM Capability | The STA sets RRCM Capability subfield to 1 to indicate support for RRCM Capability and sets to 0 if not supported. |

***3) Add a new subclause after 12.2.11 Device ID indication***

**12.2.12 Rule-based Random and Changing MAC Address (RRCM)**

**12.2.12.1 General**

To improve its privacy, a non-AP STA may desire to use a random MAC address (RMA) while still being identifiable by the same AP in subsequent associations. Rule-based Random and Changing MAC address (RRCM) allows for identification despite randomly changed MAC address at later associations., When a non-AP STA associates to an AP with one MAC address, it can still be recognized by the AP and ESS after the non-AP STA changes its MAC address before reconnecting to the same AP and ESS.

Through RRCM, a non-AP STA and AP can generate the same ‘randomized’ MAC address or addresses to be used by the non-AP STA in the next association(s) based on a common procedure through a total of three parameters. Among these parameters, two of them (Seed, Counter) are exchanged between the non-AP STA and AP, and one of them (the key – RMAK) is generated locally at both sides.

A non-AP STA and AP may generate a single RMA, which the non-AP STA can use in all message exchanges, or multiple RMAs (RMA1, RMA2 etc.), which the non-AP STA can use in different message exchanges (e.g. RMA1 in probe request frame, RMA2 in other frames).

The STA advertises the support for RRCM by setting the RRCM Capability subfield to 1 in the Extended Capabilities Element.

The generation of RMA(s) and RMAK) for RRCM are described in 12.2.12.2. The identification procedure is described in 12.2.12.3.

**12.2.12.2 RMA and Key Generation**

The procedures to generate the RMA(s) and key, RMAK, are as follows:

**RMAK** = KDF-Hash-256(KDK, "RMA Key", Min(ANonce, SNonce) || Max(ANonce, SNonce)

**RMAn** = KDF-Hash-48(RMAK, "Next RMAs", seed || n)

Where,

* KDF-Hash-256 will generate 256 bits key, RMAK. Hash is the Hash algorithm used in the AKM that the STA and AP agreed upon. KDK is derived from PTK for RRCM procedure. ANonce and SNonce are the generated values from 4-way Handshake. “RMA Key” is the string name for RMAK and is treated as an ASCII string.
* KDF-Hash-48 will generate 48-bit RMA. Seed is a 128-bit random bit string generated at non-AP STA. n is initialized with 1 and incremented by 1 until n is equal to Counter, which is the number of generated RMA(s). As an example, if three RMAs are generated, Counter=3 implies that n=1 is used to generate RMA1, n=2 is used to generate RMA2, n=3 is used to generate RMA3. The length of the counter is 16 bits, resulting in maximum 2^16 different RMA(s) generation in each association.

NOTE1-- In each association, the non-AP STA may decide to generate one or more RMA(s), where each parameter {RMAK, Seed} is re-generated and Counter is reset to one.

NOTE2-- I/G = 0 and U/L = 1 bits shall be replaced in each generated RMA, see subclause 12.2.10.

NOTE3--RMA(s) may be saved on non-AP STA and AP/ESS side until new RMA(s) are generated.
NOTE4 – When RRCM is negotiated, The PTK is partitioned into KCK, KEK, TK, and a KDK. KDK is used to derive RMAK.

**12.2.12.3 Identification Procedure**

During the association procedure, the non-AP STA and AP derive RMAK from KDK (see RMAK generation in subclause **12.2.12.2**).

Non-AP STA behaviour:

The non-AP STA initializes {Seed, Counter} values to locally generate one or more RMAs (see RMA generation in subclause **12.2.12.2**). When using FILS authentication, the non-AP STA sends the {Seed, Counter} in IE in the Association Request frame. When using FT, the non-AP STA sends the {Seed, Counter} during the initial mobility domain association in encrypted Key Data field (RRCM KDE) in the EAPOL-Key message 2/4. {Seed, Counter} is not exchanged during the FT protocol reassociations within the same ESS. For other cases, the non-AP STA sends the {Seed , Counter } in encrypted Key Data field (RRCM KDE) in the EAPOL-Key message 2/4.

AP behaviour:

After receiving {Seed, Counter} from the non-AP STA in the EAPOL-Key message 2/4 or Association Request frame in FILS authentication mode, the AP first checks the {Counter} value to determine the number of RMA(s) it needs to generate locally. The AP generates the same number of RMA(s) that non-AP STA generated (see RMA generation in subclause **12.2.12.2**).

After the non-AP STA have been disassociated, {RMAK, Seed} are deleted and {Counter} is reset to 1, while RMA(s) are stored at non-AP STA and at the (previously) associated AP or ESS.

The non-AP STA may use the generated RMAs for messaging, preparing, and establishing the next association. E.g. The non-AP STA may send to the AP with the identifiable management frame,by which the AP or ESS can identify the non-AP STA as the returned STA.

The AP or ESS can then identify the non-AP STA despite changing MAC addresses through comparison of the MAC addresses with its stored RMAs.

**12.2.12.4 The rules to use the generated RMAs**

The generated RMAs will be carried in Address 2 field of management frame sent by non-AP STA in following conditions:

a. The non-AP STA in associated state intends to send direct probe request to an AP

b. The non-AP STA intends to send authentication request and (re)association request frame to an AP

c. The non-AP STA intends to send the identifiable public action frame.

Note--1: The generated RMA should be different in the subsequent identifiable management frames once the previous identifiable management frame is successfully transmistted except authentication request and (re)association request frame.

Note--2: The STA shall not use the generated RMA in broadcast probe request with wildcast SSID

Note—3: The same RMA should be used in authentication frame and (re)association frame in once authentication and (re)association procedure.

***4) Add a new KDE to Table 12-10 KDE selectors:***

|  |
| --- |
| * KDE selectors
 |
| OUI | Data type | Meaning |
| 00-0F-AC | <ANA> | Device ID KDE |
| 00-0F-AC | <ANA> | RRCM KDE |

***5) Add the new KDE (RRCM KDE) after Figure 12-48a Device ID KDE format:***

The format of the RRCM KDE is shown in Figure 12-49 (RRCM KDE format).

|  |  |
| --- | --- |
| Seed | Counter |

 Octets 16 2

Figure 12-xx—RRCM KDE format

Seed and Counter are values to generate one or more RMA(s) through RRCM procedure. For details, see subclause **12.2.12.**

***6) Add “RRCM KDE” to 12.7.4 EAPOL-Key frame after Device ID KDE:***

 Device ID KDE is a KDE containing a device identifier

 RRCM KDE is a KDE containing {Seed, Counter} to be used for RRCM procedure

***7) Modify 12.7.6.1 General (under 12.7.6 4-way handshake):***

Message 1: Authenticator  Supplicant: EAPOL-Key(0,0,1,0,P,0,0,ANonce,0,{} or {PMKID})

Message 2: Supplicant  Authenticator: EAPOL-Key(0,1,0,0,P,0,0,SNonce,MIC,{RSNE} or {RSNE, OCI KDE} or {RSNE, RSNXE} or {RSNE, OCI KDE, RSNXE} {RSNE, Device ID KDE} or

{RSNE, OCI KDE, Device ID KDE} or {RSNE, RSNXE, Device ID KDE} or {RSNE, OCI KDE, RSNXE,

Device ID KDE}) or {RSNE, RRCM KDE} or {RSNE, OCI KDE, RRCM KDE} or {RSNE, RSNXE, RRCM KDE} or {RSNE, OCI KDE, RSNXE, RRCM KDE})

Message 3: AuthenticatorSupplicant:
EAPOL-Key(1,1,1,1,P,0,KeyRSC,ANonce,MIC,{RSNE,GTK[N]} or
{RSNE, GTK[N], OCI KDE} or {RSNE, GTK[N], RSNXE} or
{RSNE, GTK[N], OCI KDE, RSNXE}) or {RSNE, GTK[N], Device ID KDE} or {RSNE, GTK[N], OCI

KDE, Device ID KDE} or {RSNE, GTK[N], RSNXE, Device ID KDE} or {RSNE, GTK[N], OCI KDE,

RSNXE, Device ID KDE})

Message 4: Supplicant  Authenticator: EAPOL-Key(1,1,0,0,P,0,0,0,MIC,{}).

***8) Modify 12.7.6.3 4-way handshake message 2:***

Key Information:

Key Descriptor Version = 1 (ARC4 encryption with HMAC-MD5) or 2 (NIST AES key wrap with HMAC-SHA-1-128) or 3 (NIST AES key wrap with AES-128-CMAC), in all other cases 0 – same as message 1

Key Type = 1 (Pairwise) – same as message 1

Reserved = 0

Install = 0

Key Ack = 0

Key MIC = 0 when using an AEAD cipher or 1 otherwise

Secure = 0 – same as message 1

Error = 0 – same as message 1

Request = 0 – same as message 1

Encrypted Key Data = 1 when using an AEAD cipher or if the Device ID KDE is included or if RRCM KDE is included, or 0 otherwise

Reserved = 0 – unused by this protocol version

* Key Data =
* — Additionally, may include a Device ID KDE.

— Additionally, may include RRCM KDE

***9) Add new row in Table 9-62 – Association Request frame body***

|  |  |  |
| --- | --- | --- |
| **Order** | **Information** | **Notes** |
| <ANA> | Device ID | The Device ID element is optionally present when using FILS authentication; otherwise, it is not present. |
| <ANA> | RRCM | The RRCM element is present when using FILS authentication; otherwise, it is not present. |

***10) Add new row in Table 9-63 – Association Response frame body***

|  |  |  |
| --- | --- | --- |
| **Order** | **Information** | **Notes** |
| <ANA> | Device ID | The Device ID element is optionally present when using FILS authentication; otherwise, it is not present. |
| <ANA> | RRCM | The RRCM element is present when using FILS authentication; otherwise, it is not present. |

***11) Add a new row in Table 9-128 – Element IDs in 9.4.2.1 General (under 9.4.2 Elements)***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Element** | **Element ID** | **Element ID Extension** | **Extensible** | **Fragmentable** |
| Device ID (see 9.4.2.x (Device ID element)) | 255 | [ANA] | No | No |
| RRCM (see 9.4.2.296 RRCM element) | 255 | <ANA> | No | No |

***12) Add a new subclause under 9.4.2.296a (Device ID element)***

9.4.2.296b RRCM element

The RRCM element contains Seed and Counter fields that are used in RRCM procedure. The format of the RRCM element is shown in Figure 9-xxx (RRCM element format).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Element ID | Length | Element ID Extension | Seed | Counter |

Octets 1 1 1 16 2

Figure 9-xxx - RRCM element format

The Element ID, Length, and Element ID Extension fields are defined in 9.4.2.1 (General).

Seed and Counter are values to generate one or more RMA for RRCM procedure. For details, see subclause **12.2.12.**

***13) Add the following changes relevant to the use of KDK ((Proposed text modifications are based on Draft 802.11REVme\_D1.3))***

**a. (P342,line 1) 4.10.3.2 AKM operations with AS**

— If WUR frame protection is negotiated or RRCM generation is negotiated , derive a fresh WTK from the KDK

**b. (P3173,line30) under 12.6.1.1.6 PTKSA**

PTK(11ba), where the PTK includes the KDK when WUR frame protection is negotiated or RRCM is generated.

**c. (P3199,Line 64) under 12.7.1.1 General**

a) Pairwise key hierarchy, to protect individually addressed traffic(11ba), where the PTK includes a KDK if WUR frame protection is negotiated or RRCM generation is negotiated and excludes the KDK otherwise.

**d. (P3201, Line 50) under 12.7.1.3 Pairwise key hierarchy**

The PTK is partitioned into KCK, KEK, (11ba)a temporal key, and a KDK if WUR frame protection is negotiated or RRCM generation is negotiated ;otherwise the PTK is partitioned into KCK, KEK, and a temporal key. The temporal key is used by the MAC to protect individually addressed communication between the Authenticator’s and Supplicant’s respective STAs. If WUR frame protection is negotiated, the KDK is used to derive a WTK, which is used by the MAC of the WUR AP to protect and by the MAC of the WUR non-AP STA to validate individually addressed WUR Wake-up frames. PTKs are used between a single Supplicant and a single Authenticator. If RRCM generation is negotiated, the KDK is used to derive a RRMK, which is used to generate a batch of RMAs that are carried by the non-AP STA and identified by the AP.

**e. (P3202, Line 59) under 12.7.1.3 Pairwise key hierarchy**

where (11ba)Length = KCK\_bits + KEK\_bits + TK\_bits + KDK\_bits, if WUR frame protection is being negotiated or RRCM generation is being negotiated ;

**f.(P3203, Line 4) under 12.7.1.3 Pairwise key hierarchy**

(11ba)If WUR frame protection is being negotiated or RRCM generation is being negotiated, the KDK shall be computed as the next

KDK\_bits bits of the PTK:

KDK = L(PTK, KCK\_bits+KEK\_bits+TK\_bits, KDK\_bits)

Otherwise, the KDK is not derived

**g. (P3203,Line 32) under 12.7.1.3 Pairwise key hierarchy**

11ba)If WUR frame protection is negotiated, the WTK shall be derived from the KDK using the KDF

defined in 12.7.1.6.2:

WTK = KDF-Hash-Length(KDK, “WUR Temporal Key”, Min(AA,SPA) || Max(AA,SPA) ||

Min(ANonce,SNonce) || Max(ANonce,SNonce)

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 9-188 (AKM suite

selectors)).

— Length is the total number of bits to derive, i.e., number of bits of the WTK, and is equal to 128.

If RRCM is negotiated, the RRCMK shall be derived from the KDK using the KDF defined in 12.7.1.6.2. see subclause 12.2.12.2 **RMA and Key Generation**

**h. (P3211,Line 24) under 12.7.1.6.4 PMK-R1**

1ba)When WUR frame protection is negotiated or RRCM generation is negotiated, each PTK has six component keys, KCK, KEK, a

temporal key, KCK2, KEK2, and a KDK derived as follows:

(11ba)The KCK, KEK, temporal key, KCK2, and KEK2 shall be computed in the same way as when WUR frame protection is not negotiated.

(11ba)The KDK shall be computed as the next KDK\_bits bits of the PTK:

KDK = L(PTK, KCK\_bits+KEK\_bits+TK\_bits+KCK2\_bits+KEK2\_bits, KDK\_bits)

(11ba)The value of KDK\_bits is equal to the value of PMK\_bits (see 12.7.1.3 (Pairwise key hierarchy)).

**i. (insert the following change after the referenced baseline context in P3211,line 38) under 12.7.1.6.4 PMK-R1**

(11ba)If WUR frame protection is negotiated, the WTK shall be derived from the KDK using the KDF

defined in 12.7.1.6.2 (Key derivation function (KDF))):

WTK = KDF-Hash-Length(KDK, “WUR Temporal Key”, SNonce || ANonce || BSSID ||

STA-ADDR)

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 9-188 (AKM suite

selectors)).

— Length is the total number of bits to derive, i.e., number of bits of the WTK, and is equal to 128.

(11ba)The WTK is used to protect individually addressed WUR Wake-up frames, as defined in 29.10 (WUR

frame protection).

If RRCM is negotiated, the RRCMK shall be derived from the KDK using the KDF defined in 12.7.1.6.2. see subclause 12.2.12.2 **RMA and Key Generation**

**j. (P3226, line 42) under 12.7.6.2 4-way handshake message 1**

b) Derives PTK(11ba), the derived PTK including the Key derivation key (KDK) if WUR frame protection is being negotiated or RRCM generation is being negotiated .

**k. (P3269, line 54) under 12.11.2.5.3 PTKSA Key derivation with FILS authentication**

When the negotiated AKM is 00-0F-AC:16,FILS-FT is 256 bits; when the negotiated AKM is 00-0F-AC:17, FILS-FT is 384 bits; otherwise, FILS-FT is

not derived(11ba); when WUR frame protection is negotiated or RRCM generation is negotiated, the length of KDK is equal to the value of PMK\_bits (see 12.7.1.3 (Pairwise key hierarchy)); otherwise, the KDK is not derived.

**m. (P3270,line7) under 12.11.2.5.3 PTKSA Key derivation with FILS authentication**

11ba)When WUR frame protection is negotiated or RRCM generation is negotiated while doing FT initial mobility domain association using

FILS authentication,

KDK = L(PTK(#1778), ICK\_bits + KEK\_bits + TK\_bits + FILS-FT\_bits, KDK\_bits)

(11ba)When WUR frame protection is negotiated while not doing FT initial mobility domain association

using FILS authentication,

KDK = L(PTK(#1778), ICK\_bits + KEK\_bits + TK\_bits, KDK\_bits)

**n. (insert the following change after the referenced baseline context P3270,line 46) under 12.11.2.5.3 PTKSA Key derivation with FILS authentication**

11ba)If WUR frame protection is negotiated, the WTK shall be derived from the KDK using the KDF

defined in 12.7.1.6.2 (Key derivation function (KDF)):

WTK = KDF-Hash-Length(KDK, “WUR Temporal Key”, SPA || AA || SNonce || ANonce [ ||DHss ])

If RRCM is negotiated, the RRCMK shall be derived from the KDK using the KDF defined in 12.7.1.6.2. see subclause 12.2.12.2 **RMA and Key Generation**

# Proposed text change(Opt2:e-RRCM)

***1) Add following definition to 3.2***

**Rule-based Random and Changing MAC Address (RRCM):** A privacy enhancement mechanism for non-AP STA and AP to generate one or more Random Mac Addresses (RMA) for use by non-AP STA to prevent non-AP STA from being tracked (by third parties) and still allow the non-AP STA to be identified by the AP in subsequent message exchanges. “Rule-based” implies that the non-AP STA and AP apply the same procedures for generating RMA or RMA(s) locally at their sides.

**RMAK (RMA Key):** RMAK is the key that is used to generate one or more Random Mac Addresses (RMA) for RRCM procedure. This key is also used to identify unicast management frames before association.

**Pre-association identifiable Management Frame(PIMF):** The receiver can recognize the transmitter as a returned STA or a known ESS by identifying Pre-association identifiable Management Frame successfully

*2) Add following definition to 3.4*

***RPN****: RRCM packet number*

***3) Add a new capability information to Table 9-363 Extended Capabilities field***

|  |  |  |
| --- | --- | --- |
| **Bit** | **Information** | **Notes** |
| <ANA> | Device ID Support | The STA sets the Device ID Support field to 1 to indicate support for Device ID indication. Otherwise, the STA sets the Device ID field to 0. |
| <ANA> | RRCM Capability | The STA sets RRCM Capability subfield to 1 to indicate support for RRCM Capability and sets to 0 if not supported. |

***4) Add a new subclause after 12.2.11 Device ID indication***

**12.2.12 Rule-based Random and Changing MAC Address (RRCM)**

**12.2.12.1 General**

To improve its privacy, a non-AP STA may desire to use a random MAC address (RMA) while still being identifiable by the same AP in subsequent associations. Rule-based Random and Changing MAC address (RRCM) allows for identification despite randomly changed MAC address at later associations., When a non-AP STA associates to an AP with one MAC address, it can still be recognized by the AP and ESS after the non-AP STA changes its MAC address before reconnecting to the same AP and ESS.

Through RRCM, a non-AP STA and AP can generate the same ‘randomized’ MAC address or addresses to be used by the non-AP STA in the next association(s) based on a common procedure through a total of three parameters. Among these parameters, two of them (Seed, Counter) are exchanged between the non-AP STA and AP, and one of them (the key – RMAK) is generated locally at both sides.

A non-AP STA and AP may generate a single RMA along with RMAK, which the non-AP STA can use in all message exchanges, or multiple RMAs (RMA1, RMA2 etc.),along with RMAK, which the non-AP STA can use in different message exchanges (e.g. RMA1 in probe request frame, RMA2 in other frames).

The STA advertises the support for RRCM by setting the RRCM Capability subfield to 1 in the Extended Capabilities Element.

When using RMA along with RMAK in the subsequent association(s) attempt, the non-AP STA and AP can identify each other based on the exchangedPIMF.

The generation of RMA(s) and RMAK) for RRCM are described in 12.2.12.2. The identification procedure is described in 12.2.12.3. The usage of generated RMA(s) and RMAK are described in 12.2.12.4.

**12.2.12.2 RMA and Key Generation**

The procedures to generate the RMA(s) and key, RMAK, are as follows:

**RMAK** = KDF-Hash-256(KDK, "RMA Key", Min(ANonce, SNonce) || Max(ANonce, SNonce)

**RMAn** = KDF-Hash-48(RMAK, "Next RMAs", seed || n)

Where,

* KDF-Hash-256 will generate 256 bits key, RMAK. Hash is the Hash algorithm used in the AKM that the STA and AP agreed upon. KDK is derived from PTK for RRCM procedure. ANonce and SNonce are the generated values from 4-way Handshake. “RMA Key” is the string name for RMAK and is treated as an ASCII string.
* KDF-Hash-48 will generate 48-bit RMA. Seed is a 128-bit random bit string generated at non-AP STA. n is initialized with 1 and incremented by 1 until n is equal to Counter, which is the number of generated RMA(s). As an example, if three RMAs are generated, Counter=3 implies that n=1 is used to generate RMA1, n=2 is used to generate RMA2, n=3 is used to generate RMA3. The length of the counter is 16 bits, resulting in maximum 2^16 different RMA(s) generation in each association.

NOTE1-- In each association, the non-AP STA may decide to generate one or more RMA(s), where each parameter {RMAK, Seed} is re-generated and Counter is reset to one.

NOTE2-- I/G = 0 and U/L = 1 bits shall be replaced in each generated RMA, see subclause 12.2.10.

NOTE3--RMA(s) may be saved on non-AP STA and AP/ESS side until new RMA(s) are generated.
NOTE4 – When RRCM is negotiated, The PTK is partitioned into KCK, KEK, TK, and a KDK. KDK is used to derive RMAK.

**12.2.12.3 Identification Procedure**

During the association procedure, the non-AP STA and AP derive RMAK from KDK (see RMAK generation in subclause **12.2.12.2**).

Non-AP STA behaviour:

The non-AP STA may receive a frame(e.g. Beacon or association response frame) from AP to obtain the Max number of RMAs, and then initializes {Seed, Counter} values to locally generate one or more RMAs (see RMA generation in subclause **12.2.12.2**). When using FILS authentication, the non-AP STA sends the {Seed, Counter} in IE in the Association Request frame. When using FT, the non-AP STA sends the {Seed, Counter} during the initial mobility domain association in encrypted Key Data field (RRCM KDE) in the EAPOL-Key message 2/4. {Seed, Counter} is not exchanged during the FT protocol reassociations within the same ESS. For other cases, the non-AP STA sends the {Seed , Counter} in encrypted Key Data field (RRCM KDE) in the EAPOL-Key message 2/4.

Note: The Counter initiated by the non-AP STA shall not be above the Max number of RMAs indicated by the AP.

AP behaviour:

The AP may initiate the Max number of RMAs in the frame that is sent to the non-AP.

After receiving {Seed, Counter} from the non-AP STA in the EAPOL-Key message 2/4 or Association Request frame in FILS authentication mode, the AP first checks the {Counter} value to determine the number of RMA(s) it needs to generate locally. The AP generates the same number of RMA(s) that non-AP STA generated (see RMA generation in subclause **12.2.12.2**).

After the non-AP STA have been disassociated, {Seed} is deleted and {Counter} is reset to 1, while RMA(s) and RMAK are stored at non-AP STA and at the (previously) associated AP or ESS.

The non-AP STA may use the generated RMAs and RMAK for messaging, preparing, and establishing the next association. E.g. The non-AP STA may send the PIMF to the AP (see subclause 12.2.12.6 PIMF transmission) ,by which the AP or ESS can identify the non-AP STA as the returned STA after mapping the RMA to the stored RMAK and verifying MIC in the VIE element successfully (see subclause 12.2.12.6 PIMF reception)).

After receving the PIMF,The AP may make a respond with a PIMF(such as ANQP response frame to the ANQP request frame), in which the VIE element is carried in the framebody and the same received RMA in A1 field iscarried in the MAC header

**12.2.12.4 The rules to use the generated RMAs and RMAK**

The generated RMAs will be carried in Address 2 field of MAC header,along with VIE element in the frame body of PIMF sent by STA in following conditions:

a. The non-AP STA in associated state intends to send direct probe request to an AP

b. The non-AP STA intends to send authentication request and (re)association request frame to an AP

c. The non-AP STA intends to send the identifiable public action frame.

d. The AP sends the corresponding response frame according to the rules defined in subclause 11.1.4.3.4 (Criteria for sending a response), subclause 11.3.4 (Authentication and deauthentication) and subclause ,subclause (11.3.5 Association, reassociation, and disassociation) and subclause (11.22.3.3 ANQP procedures).

Note--1: The generated RMA should be different in the subsequent PIMF once the previous one is successfully transmistted except authentication request and (re)association request frame.

Note--2: The STA shall not use the generated RMA in broadcast probe request with wildcast SSID

Note—3: The same RMA should be used in authentication frame and (re)association frame in a single authentication and (re)association attempt.

**12.2.12.5 PIMF AAD**

The PIMF Additional Authentication Data (AAD) is constructed from the MPDU header. AAD construction is

performed as follows:

a) FC—MPDU Frame Control field, with the following modifications:

 1) Retry subfield (bit 11) masked to 0

 2) Power Management subfield (bit 12) masked to 0

 3) More Data subfield (bit 13) masked to 0

 4) Other subfields are not modified

b) A1—MPDU Address 1 field.

c) A2—MPDU Address 2 field.

d) A3—MPDU Address 3 field.

Figure 12-xx (PIMF AAD construction) depicts the format of the AAD. The length of the AAD is 20 octets.

|  |  |  |  |
| --- | --- | --- | --- |
| FC | A1 | A2 | A3 |

Octets: 2 6 6 6

 Figure 12-XX—PIMF AAD construction

**12.2.12.5 PIMF transmission**

When a STA transmits an PIMF, it shall

a) Select the <RMA,RMAK> peer currently active for transmission of frame to the intended recipients and construct MAC header and VIE(see subclause 9.4.2.296c VIE element) with the MIC field masked to 0 respectively. the transmitting STA shall insert a strictly increasing integer into the VIE RPN field.

b) Compute AAD as specified in 12.2.15.5(PIBF AAD construction).

c) Compute an integrity value over the concatenation of AAD and the management frame body

including VIE, and the Timestamp field masked to 0 if it is a identifiable probe response frame, and insert

the output into the VIE MIC field. The integrity value is 64 bits and is computed using AES-128-CMAC;

d) Compose the frame as the IEEE 802.11 header, management frame body, including VIE, and FCS. The VIE shall appear last in the frame body.

e) Transmit the frame.

Once a STA transmits a PIMF using a new RMAK, the STA shall not transmit PIMF using the previous(#1847) RMAK.

**12.2.12.6 PIMF reception**

When a STA with RRCMactivite equal to True receives a PIMF protected by AES-128-CMAC, it shall

a) Identify the appropriate <RMA,RMAK >and unassociated state. If the frame is a PIMF and no such <RMA,RMAK> peer exists, terminate RRCM identification processing for this reception. Otherwise, process the frame.

b) Perform replay protection on the received frame. The receiver shall interpret the VIE RPN field as a 48-bit unsigned integer.

c) If the frame is a PIMF, the receiver shall compare this VIE RPN value to the value of the (#1504)replay counter for the RMAK. If the value from the received VIE RPN field is less than or equal to the replay counter value for this RMAK, the receiver shall discard the frame and increment the dot11CMACReplays counter by 1.

d) Compute AAD for this Management frame, as specified in 12.2.12.5 (PIMF AAD construction).

e) Extract and save the received MIC value, and compute a verifier over the concatenation of AAD, the

management frame body, with the Timestamp field masked to 0 if it is an identifiable probe response frame, and VIE, with the MIC field masked to 0 in the VIE. the integrity value is 64 bits and is computed using AES-128-CMAC; If the result does not match the received MIC value, then the receiver shall discard the frame, increment the dot11RSNAStatsBIPMICErrors counter by 1, and terminate RRCM indentification processing for this reception.

**12.2.12.7 RPN and replay detection**

if RRCMActivated is true, the recipient shall maintain a 48-bit replay counter for each RMAK. The transmitter shall maintain a single RPN for each RMAK.

The RPN shall be implemented as a 48-bit strictly increasing integer, initialized to 1 when the corresponding RMAK is initialized. The transmitter may reinitialize the sequence counter when the RMAK is refreshed.

See 12.2.12.5 (PIMF transmission) and 12.2.12.6 (PIMF reception) for per frame PIMF processing.

***5) Add a new KDE to Table 12-10 KDE selectors:***

|  |
| --- |
| * KDE selectors
 |
| OUI | Data type | Meaning |
| 00-0F-AC | <ANA> | Device ID KDE |
| 00-0F-AC | <ANA> | RRCM KDE |

***6) Add the new KDE (RRCM KDE) after Figure 12-48a Device ID KDE format:***

The format of the RRCM KDE is shown in Figure 12-49 (RRCM KDE format).

|  |  |
| --- | --- |
| Seed | Counter |

 Octets 16 2

Figure 12-xx—RRCM KDE format

Seed and Counter are values to generate one or more RMA(s) through RRCM procedure. For details, see subclause **12.2.12.**

***7) Add “RRCM KDE” to 12.7.4 EAPOL-Key frame after Device ID KDE:***

 Device ID KDE is a KDE containing a device identifier

 RRCM KDE is a KDE containing {Seed, Counter} to be used for RRCM procedure

***8) Modify 12.7.6.1 General (under 12.7.6 4-way handshake):***

Message 1: Authenticator  Supplicant: EAPOL-Key(0,0,1,0,P,0,0,ANonce,0,{} or {PMKID})

Message 2: Supplicant  Authenticator: EAPOL-Key(0,1,0,0,P,0,0,SNonce,MIC,{RSNE} or {RSNE, OCI KDE} or {RSNE, RSNXE} or {RSNE, OCI KDE, RSNXE} {RSNE, Device ID KDE} or

{RSNE, OCI KDE, Device ID KDE} or {RSNE, RSNXE, Device ID KDE} or {RSNE, OCI KDE, RSNXE,

Device ID KDE}) or {RSNE, RRCM KDE} or {RSNE, OCI KDE, RRCM KDE} or {RSNE, RSNXE, RRCM KDE} or {RSNE, OCI KDE, RSNXE, RRCM KDE})

Message 3: AuthenticatorSupplicant:
EAPOL-Key(1,1,1,1,P,0,KeyRSC,ANonce,MIC,{RSNE,GTK[N]} or
{RSNE, GTK[N], OCI KDE} or {RSNE, GTK[N], RSNXE} or
{RSNE, GTK[N], OCI KDE, RSNXE}) or {RSNE, GTK[N], Device ID KDE} or {RSNE, GTK[N], OCI

KDE, Device ID KDE} or {RSNE, GTK[N], RSNXE, Device ID KDE} or {RSNE, GTK[N], OCI KDE,

RSNXE, Device ID KDE})

Message 4: Supplicant  Authenticator: EAPOL-Key(1,1,0,0,P,0,0,0,MIC,{}).

***9) Modify 12.7.6.3 4-way handshake message 2:***

Key Information:

Key Descriptor Version = 1 (ARC4 encryption with HMAC-MD5) or 2 (NIST AES key wrap with HMAC-SHA-1-128) or 3 (NIST AES key wrap with AES-128-CMAC), in all other cases 0 – same as message 1

Key Type = 1 (Pairwise) – same as message 1

Reserved = 0

Install = 0

Key Ack = 0

Key MIC = 0 when using an AEAD cipher or 1 otherwise

Secure = 0 – same as message 1

Error = 0 – same as message 1

Request = 0 – same as message 1

Encrypted Key Data = 1 when using an AEAD cipher or if the Device ID KDE is included or if RRCM KDE is included, or 0 otherwise

Reserved = 0 – unused by this protocol version

* Key Data =
* — Additionally, may include a Device ID KDE.

— Additionally, may include RRCM KDE

***10) Add new row in Table 9-62 – Association Request frame body***

|  |  |  |
| --- | --- | --- |
| **Order** | **Information** | **Notes** |
| <ANA> | Device ID | The Device ID element is optionally present when using FILS authentication; otherwise, it is not present. |
| <ANA> | RRCM | The RRCM element is present when using FILS authentication; otherwise, it is not present. |

***11) Add new row in Table 9-63 – Association Response frame body***

|  |  |  |
| --- | --- | --- |
| **Order** | **Information** | **Notes** |
| <ANA> | Device ID | The Device ID element is optionally present when using FILS authentication; otherwise, it is not present. |
| <ANA> | RRCM | The RRCM element is present when using FILS authentication; otherwise, it is not present. |

***12) Add a new row in Table 9-128 – Element IDs in 9.4.2.1 General (under 9.4.2 Elements)***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Element** | **Element ID** | **Element ID Extension** | **Extensible** | **Fragmentable** |
| Device ID (see 9.4.2.x (Device ID element)) | 255 | [ANA] | No | No |
| RRCM (see 9.4.2.296 RRCM element) | 255 | <ANA> | No | No |

***13) Add a new subclause under 9.4.2.296a (Device ID element)***

9.4.2.296b RRCM element

The RRCM element contains Seed and Counter fields that are used in RRCM procedure. The format of the RRCM element is shown in Figure 9-xxx (RRCM element format).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Element ID | Length | Element ID Extension | Seed | Counter |

Octets 1 1 1 16 2

Figure 9-xxx - RRCM element format

The Element ID, Length, and Element ID Extension fields are defined in 9.4.2.1 (General).

Seed and Counter are values to generate one or more RMA for RRCM procedure. For details, see subclause **12.2.12.**

9.4.2.296c VIE element

|  |  |  |  |
| --- | --- | --- | --- |
| Element ID | Length | RPN | MIC |

Octets 1 1 6 8

The Element ID and Length fields are defined in 9.4.2.1 (General).

The RPN field is used to detect replay of the identifiable management frame.

The MIC field contains a message integrity code calculated over PIMF as specified in 12.2.12.6 (PIMF transmission) and 12.2.12.7 (PIMF reception). The length of the

MIC field depends on the specific cipher negotiated and is either 8 octets (for AES-128-CMAC)

***14) Add the following changes relevant to the use of KDK ((Proposed text modifications are based on Draft 802.11REVme\_D1.3))***

**a. (P342,line 1) 4.10.3.2 AKM operations with AS**

— If WUR frame protection is negotiated or RRCM generation is negotiated , derive a fresh WTK from the KDK

**b. (P3173,line30) under 12.6.1.1.6 PTKSA**

PTK(11ba), where the PTK includes the KDK when WUR frame protection is negotiated or RRCM is generated.

**c. (P3199,Line 64) under 12.7.1.1 General**

a) Pairwise key hierarchy, to protect individually addressed traffic(11ba), where the PTK includes a KDK if WUR frame protection is negotiated or RRCM generation is negotiated and excludes the KDK otherwise.

**d. (P3201, Line 50) under 12.7.1.3 Pairwise key hierarchy**

The PTK is partitioned into KCK, KEK, (11ba)a temporal key, and a KDK if WUR frame protection is negotiated or RRCM generation is negotiated ;otherwise the PTK is partitioned into KCK, KEK, and a temporal key. The temporal key is used by the MAC to protect individually addressed communication between the Authenticator’s and Supplicant’s respective STAs. If WUR frame protection is negotiated, the KDK is used to derive a WTK, which is used by the MAC of the WUR AP to protect and by the MAC of the WUR non-AP STA to validate individually addressed WUR Wake-up frames. PTKs are used between a single Supplicant and a single Authenticator. If RRCM generation is negotiated, the KDK is used to derive a RRMK, which is used to generate a batch of RMAs that are carried by the non-AP STA and identified by the AP.

**e. (P3202, Line 59) under 12.7.1.3 Pairwise key hierarchy**

where (11ba)Length = KCK\_bits + KEK\_bits + TK\_bits + KDK\_bits, if WUR frame protection is being negotiated or RRCM generation is being negotiated ;

**f.(P3203, Line 4) under 12.7.1.3 Pairwise key hierarchy**

(11ba)If WUR frame protection is being negotiated or RRCM generation is being negotiated, the KDK shall be computed as the next

KDK\_bits bits of the PTK:

KDK = L(PTK, KCK\_bits+KEK\_bits+TK\_bits, KDK\_bits)

Otherwise, the KDK is not derived

**g. (P3203,Line 32) under 12.7.1.3 Pairwise key hierarchy**

11ba)If WUR frame protection is negotiated, the WTK shall be derived from the KDK using the KDF

defined in 12.7.1.6.2:

WTK = KDF-Hash-Length(KDK, “WUR Temporal Key”, Min(AA,SPA) || Max(AA,SPA) ||

Min(ANonce,SNonce) || Max(ANonce,SNonce)

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 9-188 (AKM suite

selectors)).

— Length is the total number of bits to derive, i.e., number of bits of the WTK, and is equal to 128.

If RRCM is negotiated, the RRCMK shall be derived from the KDK using the KDF defined in 12.7.1.6.2. see subclause 12.2.12.2 **RMA and Key Generation**

**h. (P3211,Line 24) under 12.7.1.6.4 PMK-R1**

1ba)When WUR frame protection is negotiated or RRCM generation is negotiated, each PTK has six component keys, KCK, KEK, a

temporal key, KCK2, KEK2, and a KDK derived as follows:

(11ba)The KCK, KEK, temporal key, KCK2, and KEK2 shall be computed in the same way as when WUR frame protection is not negotiated.

(11ba)The KDK shall be computed as the next KDK\_bits bits of the PTK:

KDK = L(PTK, KCK\_bits+KEK\_bits+TK\_bits+KCK2\_bits+KEK2\_bits, KDK\_bits)

(11ba)The value of KDK\_bits is equal to the value of PMK\_bits (see 12.7.1.3 (Pairwise key hierarchy)).

**i. (insert the following change after the referenced baseline context in P3211,line 38) under 12.7.1.6.4 PMK-R1**

(11ba)If WUR frame protection is negotiated, the WTK shall be derived from the KDK using the KDF

defined in 12.7.1.6.2 (Key derivation function (KDF))):

WTK = KDF-Hash-Length(KDK, “WUR Temporal Key”, SNonce || ANonce || BSSID ||

STA-ADDR)

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 9-188 (AKM suite

selectors)).

— Length is the total number of bits to derive, i.e., number of bits of the WTK, and is equal to 128.

(11ba)The WTK is used to protect individually addressed WUR Wake-up frames, as defined in 29.10 (WUR

frame protection).

If RRCM is negotiated, the RRCMK shall be derived from the KDK using the KDF defined in 12.7.1.6.2. see subclause 12.2.12.2 **RMA and Key Generation**

**j. (P3226, line 42) under 12.7.6.2 4-way handshake message 1**

b) Derives PTK(11ba), the derived PTK including the Key derivation key (KDK) if WUR frame protection is being negotiated or RRCM generation is being negotiated .

**k. (P3269, line 54) under 12.11.2.5.3 PTKSA Key derivation with FILS authentication**

When the negotiated AKM is 00-0F-AC:16,FILS-FT is 256 bits; when the negotiated AKM is 00-0F-AC:17, FILS-FT is 384 bits; otherwise, FILS-FT is

not derived(11ba); when WUR frame protection is negotiated or RRCM generation is negotiated, the length of KDK is equal to the value of PMK\_bits (see 12.7.1.3 (Pairwise key hierarchy)); otherwise, the KDK is not derived.

**m. (P3270,line7) under 12.11.2.5.3 PTKSA Key derivation with FILS authentication**

11ba)When WUR frame protection is negotiated or RRCM generation is negotiated while doing FT initial mobility domain association using

FILS authentication,

KDK = L(PTK(#1778), ICK\_bits + KEK\_bits + TK\_bits + FILS-FT\_bits, KDK\_bits)

(11ba)When WUR frame protection is negotiated while not doing FT initial mobility domain association

using FILS authentication,

KDK = L(PTK(#1778), ICK\_bits + KEK\_bits + TK\_bits, KDK\_bits)

**n. (insert the following change after the referenced baseline context P3270,line 46) under 12.11.2.5.3 PTKSA Key derivation with FILS authentication**

11ba)If WUR frame protection is negotiated, the WTK shall be derived from the KDK using the KDF

defined in 12.7.1.6.2 (Key derivation function (KDF)):

WTK = KDF-Hash-Length(KDK, “WUR Temporal Key”, SPA || AA || SNonce || ANonce [ ||DHss ])

If RRCM is negotiated, the RRCMK shall be derived from the KDK using the KDF defined in 12.7.1.6.2. see subclause 12.2.12.2 **RMA and Key Generation**

# Proposed text change(Opt3: RRCM+MAAD)

802.11 bh Draft 0.2 is base

*Add the following definitions to 3.2*

**Rule-based Random and Changing MAC Address (RRCM):** A privacy enhancement mechanism for non-AP STA and AP to generate one or more Random Mac Addresses (RMA) for use by non-AP STA in order to prevent non-AP STA from being tracked (by third parties) and still allow the non-AP STA to be identified by the AP in subsequent message exchanges. “Rule-based” implies that the non-AP STA and AP apply the same procedures for generating RMA or RMAs locally at their sides.

**RMAK (RMA Key):** RMAK is the key that is used to generate one or more Random Mac Addresses (RMA) for RRCM procedure

*Add following Acronym to 3.4.*

MAAD MAC Address Designation

*At 4.5.4.10, edit last sentence to read*

Such a STA, when reconnecting to a network, can opt-in to exchange a device identifier that allows the network to recognize the device and/or use a MAC address that has been allocated by the network or the STA, whilst still protecting the information from third parties.

***Clause 6.3***

***We might need an “MLME-RCM” primitive so that the SME can instruct the MLME to set up which schemes (device ID, MAAD, RRCM the STA will support. Work in ARC and TGme will probably change the way this is written. The primitive will consist of a single MLME-RCM.request. Discussions in TGbh to decide if MIB or MLME.***

*At 9.3.3.5 Association Request frame format*

*Insert new row in Table 9-62 Association Request frame body P23*

|  |  |  |
| --- | --- | --- |
| **Order** | **Information** | **Notes** |
| <ANA> | Device ID | The Device ID element is optionally present when using FILS authentication; otherwise, it is not present. |
| <ANA> | MAAD | The MAAD element is optionally present when using FILS authentication; otherwise, it is not present |
| <ANA> | RRCM | The RRCM element is present when using FILS authentication; otherwise, it is not present. |

*At 9.3.3.6 Association Response frame format*

*Insert new row in Table 9-63 Association Response frame body P1031*

|  |  |  |
| --- | --- | --- |
| **Order** | **Information** | **Notes** |
| <ANA> | Device ID | The Device ID element is optionally present when using FILS authentication; otherwise, it is not present. |
| <ANA> | MAAD | The MAAD element is optionally present when using FILS authentication; otherwise, it is not present |
| <ANA> | RRCM | The RRCM element is present when using FILS authentication; otherwise, it is not present. |

*Insert new rows in Table 9-64 Reassociation Request frame body*

|  |  |  |
| --- | --- | --- |
| **Order** | **Information** | **Notes** |
| <ANA> | MAAD | The MAAD element is optionally present when using FILS authentication; otherwise, it is not present |
| <ANA> | RRCM | The RRCM element is present when using FILS authentication; otherwise, it is not present. |

*Insert new rows in Table 9-65 Reassociation Response frame body*

|  |  |  |
| --- | --- | --- |
| **Order** | **Information** | **Notes** |
| <ANA> | MAAD | The MAAD element is optionally present when using FILS authentication; otherwise, it is not present |
| <ANA> | RRCM | The RRCM element is present when using FILS authentication; otherwise, it is not present. |

*At 9.4.2.1 Insert new rows in Table 9-128 Element IDs P23*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Element | Element ID | Element ID Extension | Extensible | Fragmentable |
| Device ID (see 9.4.2.x (Device ID element)) | 255 | <ANA> | No | No |
| MAAD (see 9.4.2.xx MAAD element) | 255 | <ANA> | No | No |
| RRCM (see 9.4.2.xxxx RRCM element) | 255 | <ANA> | No | No |

*At 9.4.2.241 Insert new rows in Table 9-363 Extended Capabilities field, P24*

|  |  |  |
| --- | --- | --- |
| **Bit** | **Information** | **Notes** |
| <ANA> | Device ID support | The STA sets the Device ID Support field to 1 to indicate support for Device ID indication. Otherwise, the STA sets the Device ID field to 0. |
| <ANA> | MAAD Capability | A STA sets MAAD Capability subfield to 1 to indicate support for MAAD and sets to 0 if MAAD is not supported. |
| <ANA> | RRCM Capability | The STA sets RRCM Capability subfield to 1 to indicate support for RRCM Capability and sets to 0 if not supported. |

*Insert following subclauses after 9.4.2.296a “Device ID element” P 24*

9.4.2.x MAAD element

The MAAD element contains a MAAD MAC address. The format of the MAAD element is shown in Figure 9-y.

|  |  |  |  |
| --- | --- | --- | --- |
| Element ID | Length | Element ID Extension | MAAD MAC |

 Octets 1 1 1 6

**Figure 9-y MAAD element**

The Element ID, Length, and Element ID Extension fields are defined in 9.4.2.1 (General).

The MAAD MAC field is a 48-bit MAC address.

**9.4.2.xx RRCM element**

The RRCM element contains Seed and Counter fields that are used in RRCM procedure. The format of the RRCM element is shown in Figure 9-xxx (RRCM element format).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Element ID | Length | Element ID Extension | Seed | Counter |

Octets 1 1 1 16 2

Figure 9-xxx - RRCM element format

The Element ID, Length, and Element ID Extension fields are defined in 9.4.2.1 (General).

Seed and Counter are values to generate one or more RMA for RRCM procedure. For details, see subclause **12.2.11.4**

**12. Security**

*Add the following new subclause after 12.2.10*

**12.2.11 Mitigation of random and changing MAC address**

To mitigate tracking and traffic analysis, a non-AP STA may randomly change its MAC address (see 4.5.4.10). For some services, however, it may be desirable to the user that the non-AP STA is identified by the AP and network services. Device ID indication, MAAD and RRCM may be used to identify the non-AP STA whilst still being unidentifiable to a third party.

When using device ID indication, an AP may provide a device ID, contained in a device ID KDE in EAPOL Key-message 3 of the 4-way handshake, to a non-AP STA and the non-AP STA may provide that same device ID, in a device ID KDE in EAPOL Key-message 2 of the 4-way handshake, to any AP in the same ESS to allow the network to recognize the same non-AP STA when it returns to the ESS even if it changes its MAC address.

When using MAAD, an AP may provide a random MAC address (MAAD MAC address) contained in a MAAD KDE in EAPOL Key-message 3 of the 4-way handshake, to a non-AP STA when it associates, and the non-AP STA may then use that MAAD MAC address as its TA when associating the next time to that ESS or AP. Hence, that AP or ESS can also recognize that non-AP STA pre-association.

When using RRCM, a non-AP STA and AP generate the same non-AP STA Random MAC address or addresses (RMA(s)) to be used in the next association(s) through following the same procedure. The non-AP STA can use the RMA(s) in its next association(s) and will be identified by the AP.

A non-AP STA and an AP may indicate support for device ID indication, MAAD, or RRCM either individually or in combination. An MLME-RCM.request may be used to set which scheme(s) the STA supports.

No combination of MAAD or RRCM can be used at the same time. If the AP and non-AP STA indicate support for MAAD, and RRCM then if the non-AP STA includes an RRCM KDE in EAPOL Key-message 2 of the 4-way handshake, then the AP shall not include a MAAD KDE in EAPOL-Key message 3 of the 4-way handshake.

***Renumber Device ID indication clause 12.2.11 as 12.2.11.1. Then delete the first paragraph and retain the rest (with changes as appropriate from the CID resolutions)***

**12.2.11.1 Device ID indication**

~~An AP may provide an identifier to a non-AP STA and the non-AP STA may opt-in to providing that identifier to any AP in the same ESS to allow the network to recognize the same non-AP STA when it returns to the ESS even if it changes its MAC address. Exchanges of this identifier information are protected from third parties to limit the tracking capability to the APs in an ESS~~.

***Following existing text subject to change from CIDs***

A non-AP STA indicates support for this capability in the Device ID Support subfield in the Extended RSN Capabilities field (see 9.4.2.241 (RSN Extension Element)). An AP shall not send an identifier to a non-AP STA that does not indicate support for this capability.

When using FILS authentication, the non-AP STA sends the identifier, if it has one and opts-in to using it, in the Association Request frame and the AP sends a new identifier in the Association Response frame. When using FT, the non-AP STA sends the identifier, if it has one and opts-in to using it, during the initial mobility domain association the EAPOL-Key message 2/4 and the AP sends a new identifier in the EAPOL-Key message 3/4; the identifier or a new identifier are not exchanged during the FT protocol reassociations within the same ESS. For other cases, the non-AP STA sends the identifier, if it has one and opts-in to using it, during the initial 4-way handshake in the EAPOL-Key message 2/4 and the AP sends a new identifier in the EAPOL-Key message 3/4. When the non-AP STA sends the opaque identifier, it shall send the most recently received value from an AP in the ESS without modification.

***Insert following sub clauses 12.2.11.2 and 12.2.11.3***

**12.2.11.2 MAC Address Designation (MAAD) operation**

A STA advertises support for MAAD by setting the MAAD Capability subfield to 1 in the Extended Capabilities element in Probe Response, Association Response and Reassociation Response frames.

Each time the non-AP STA associates to the AP/ESS, it receives a new MAAD MAC address during the RSN association. The non-AP STA may then use that MAAD MAC address as its TA the next time it probes or requests association to that same AP/ESS.

When the associating non-AP STA advertises support for MAAD, the AP may allocate a new MAAD MAC address to the non-AP STA by including a MAAD KDE in EAPOL-Key message 3 of 4-way handshake, or, when using FILS authentication, including the MAAD element in the Association Response frame.

The non-AP STA should store that newly allocated MAAD MAC as an identifier for that AP/ESS. The non-AP STA then may use that allocated MAAD MAC address as its TA when it next associates to that same AP or another AP in the same ESS. In so doing, the AP/ESS will identify the non-AP STA. When reassociating to the same AP or another AP in the same ESS, the non-AP STA uses the MAAD MAC address that it used for the association.

Note 1: Allocating a new MAAD MAC during each association ensures that the non-AP STA will use a different TA for each association and hence that non-AP STA is unidentifiable to a third party.

The MAAD MAC address is a 48-bit address that is constructed from the locally administered address space (see 12.2.10). The non-AP STA may then store this address and use it as the TA in the next association request to that same AP. An AP should generate the MAAD MAC addresses on a random basis such that a returning non-AP STA cannot be identified by a third party from the TA it is using. A list of MAAD MACs and respective non-AP STAs shall be stored by the AP and used as an identifier for each non-AP STA. A non-AP STA should store the latest MAAD MAC received from a particular AP such that the next time the non-AP STA associates to that AP, the AP can identify the non-AP STA.

When a non-AP STA sends an Association Request using an allocated MAAD MAC address as the TA, to the AP that allocated that address, then that AP can identify the non-AP STA before association is started or completed. A non-AP STA should use a random MAC address when sending Probe Requests. A non-AP STA that has been allocated a MAAD MAC address, may use that address when directly probing the AP or ESS that allocated that address when directed by the AP or ESS such that, for example, the ESS may steer the non-AP STA to an appropriate AP. Such steering applications are outside of scope. A non-AP STA that has been allocated a MAAD MAC address, may use that address in an ANQP packet so that the AP that allocated that MAAD MAC may identify the non-AP STA, i.e., the non-AP STA had previously associated with that AP.

**12.2.11.3 Rule-based Random and Changing MAC Address (RRCM)**

**12.2.11.3.1 General**

To improve its privacy, a non-AP STA may desire to use a random MAC address (RMA) while still being identifiable by the same AP in subsequent associations. Rule-based Random and Changing MAC address (RRCM) allows for identification despite randomly changed MAC address at later associations., When a non-AP STA associates to an AP with one MAC address, it can still be recognized by the AP and ESS after the non-AP STA changes its MAC address before reconnecting to the same AP and ESS.

Through RRCM, a non-AP STA and AP can generate the same ‘randomized’ MAC address or addresses to be used by the non-AP STA in the next association(s) based on a common procedure through a total of three parameters. Among these parameters, two of them (Seed, Counter) are exchanged between the non-AP STA and AP, and one of them (the key – RMAK) is generated locally at both sides.

A non-AP STA and AP may generate a single RMA, which the non-AP STA can use in all message exchanges, or multiple RMAs (RMA1, RMA2 etc.), which the non-AP STA can use in different message exchanges (e.g., RMA1 in probe request frame, RMA2 in other frames).

The STA advertises the support for RRCM by setting the RRCM Capability subfield to 1 in the Extended Capabilities Element.

The generation of RMA(s) and RMAK) for RRCM are described in 12.2.11.4.2. The identification procedure is described in 12.2.11.4.3.

**12.2.11.3.2 RMA, Keys, and Tag Generation**

The procedures to generate the RMA(s) and key, RMAK, are as follows:

**RMAK** = KDF-Hash-256(KDK, "RMA Key", Min(ANonce, SNonce) || Max(ANonce, SNonce)

**RMAn** = KDF-Hash-48(RMAK, "Next RMAs", seed || n)

Where,

* KDF-Hash-256 will generate 256 bits key, RMAK. Hash is the Hash algorithm used in the AKM that the STA and AP agreed upon. KDK is derived from PTK for RRCM procedure. ANonce and SNonce are the generated values from 4-way Handshake. “RMA Key” is the string name for RMAK and is treated as an ASCII string.
* KDF-Hash-48 will generate 48-bit RMA. Seed is a 128-bit random bit string generated at non-AP STA. n is initialized with 1 and incremented by 1 until n is equal to Counter, which is the number of generated RMA(s). As an example, if three RMAs are generated, Counter=3 implies that n=1 is used to generate RMA1, n=2 is used to generate RMA2, n=3 is used to generate RMA3. The length of the counter is 16 bits, resulting in maximum 2^16 different RMA(s) generation in each association.

NOTE1-- In each association, the non-AP STA may decide to generate one or more RMA(s), where each parameter {RMAK, Seed} is re-generated and Counter is reset to one.

NOTE2-- I/G = 0 and U/L = 1 bits shall be replaced in each generated RMA, see subclause 12.2.10.

NOTE3--RMA(s) may be saved on non-AP STA and AP/ESS side until new RMA(s) are generated.
NOTE4 – When RRCM is negotiated, The PTK is partitioned into KCK, KEK, TK, and a KDK. KDK is used to derive RMAK.

**12.2.11.3.3 Identification Procedure**

During the association procedure, the non-AP STA and AP derive RMAK from KDK (see RMAK generation in subclause **12.2.11.4.2**).

Non-AP STA behaviour:

The non-AP STA initializes {Seed, Counter} values to locally generate one or more RMAs (see RMA generation in subclause **12.2.11.4.2**). When using FILS authentication, the non-AP STA sends the {Seed, Counter} in IE in the Association Request frame. When using FT, the non-AP STA sends the {Seed, Counter} during the initial mobility domain association in encrypted Key Data field (RRCM KDE) in the EAPOL-Key message 2 of 4-way handshake. {Seed, Counter} is not exchanged during the FT protocol reassociations within the same ESS. For other cases, the non-AP STA sends the {Seed , Counter } in encrypted Key Data field (RRCM KDE) in the EAPOL-Key message 2 of 4-way handshake.

AP behaviour:

After receiving {Seed, Counter} from the non-AP STA in the EAPOL-Key message 2 of 4-way handshake or Association Request frame in FILS authentication mode, the AP first checks the {Counter} value to determine the number of RMA(s) it needs to generate locally. The AP generates the same number of RMA(s) that non-AP STA generated (see RMA generation in subclause **12.2.11.2**).

After the non-AP STA have been disassociated, {RMAK, Seed} are deleted and {Counter} is reset to 1, while RMA(s) are stored at non-AP STA and at the (previously) associated AP or ESS.

The non-AP STA may use the generated RMAs for messaging, preparing, and establishing the next association. E.g. The non-AP STA may send to the AP with the identifiable management frame,by which the AP or ESS can identify the non-AP STA as the returned STA.

The AP or ESS can then identify the non-AP STA despite changing MAC addresses through comparison of the MAC addresses with its stored RMAs.

* EAPOL-Key frames

*Add a new row into Table 12-10 (KDE selectors) P26 as shown below:*

|  |
| --- |
| * KDE selectors
 |
| OUI | Data type | Meaning |
| 00-0F-AC | <ANA> | Device ID KDE |
| 00-0F-AC | <ANA> | MAAD KDE |
| 00-0F-AC | <ANA> | RRCM KDE |

*Make following additions for the new KDE at the end of 12.7.2 as shown below:*

The format of the MAAD KDE is shown in Figure 12-48b (MAAD KDE format).

|  |
| --- |
| MAADMAC |

 Octets 6

Figure 12-48b—MAAD KDE format

The MAAD MAC field contains a MAC address.

The format of the RRCM KDE is shown in Figure 12-49 (RRCM KDE format).

|  |  |
| --- | --- |
| Seed | Counter |

 Octets 16 2

Figure 12-49—RRCM KDE format

Seed and Counter are values to generate one or more RMA(s) through RRCM procedure. For details, see subclause **12.2.11.3.**

* EAPOL-Key frame notation

*Insert following text after OCI KDE (shown for reference)*

 OCI KDE is a KDE containing operating channel information

 Device ID KDE is a KDE containing a device identifier

 MAAD KDE is a KDE containing a MAAD MAC

 RRCM KDE is a KDE containing {Seed, Nonce, Counter, Tag} to be used for RRCM procedure

* 4-way handshake
* General

*Modify 12.7.6.1 P27 as shown below:*

RSNA defines a protocol using EAPOL-Key frames called the *4-way handshake*. The handshake completes the IEEE 802.1X authentication process. The information flow of the 4-way handshake is as follows:

Message 1: Authenticator  Supplicant: EAPOL-Key(0,0,1,0,P,0,0,ANonce,0,{} or {PMKID})

Message 2: Supplicant  Authenticator: EAPOL-Key(0,1,0,0,P,0,0,SNonce,MIC,{RSNE} or {RSNE, OCI KDE} or {RSNE, RSNXE} or {RSNE, OCI KDE, RSNXE} or

{RSNE, OCI KDE, RSNXE} or {RSNE, Device ID KDE, RRCM KDE} or {RSNE, OCI KDE, Device ID KDE, RRCM KDE} or {RSNE, RSNXE, Device ID KDE, RRCM KDE} or {RSNE, OCI KDE, RSNXE, Device ID KDE, RRCM KDE} or

{RSNE, OCI KDE, RSNXE} or {RSNE, RRCM KDE} or {RSNE, OCI KDE, RRCM KDE} or {RSNE, RSNXE, RRCM KDE} or {RSNE, OCI KDE, RSNXE, RRCM KDE})

Message 3: AuthenticatorSupplicant:
EAPOL-Key(1,1,1,1,P,0,KeyRSC,ANonce,MIC,{RSNE,GTK[N]} or
{RSNE, GTK[N], OCI KDE} or {RSNE, GTK[N], RSNXE} or
{RSNE, GTK[N], OCI KDE, RSNXE} or
{RSNE, GTK[N], Device ID KDE} or {RSNE, GTK[N], OCI KDE, Device ID KDE} or
{RSNE, GTK[N], RSNXE, Device ID KDE} or {RSNE, GTK[N], OCI KDE, RSNXE, Device ID KDE} or
{RSNE, GTK[N], MAAD KDE} or {RSNE, GTK[N], OCI KDE, MAAD KDE} or
{RSNE, GTK[N], RSNXE, MAAD KDE} or {RSNE, GTK[N], OCI KDE, RSNXE, MAAD KDE} or
{RSNE, GTK[N], MAAD KDE} or {RSNE, GTK[N], OCI KDE, MAAD KDE} or
{RSNE, GTK[N], RSNXE, Device ID, MAAD KDE} or
{RSNE, GTK[N], OCI KDE, RSNXE, Device ID, MAAD KDE})

Message 4: Supplicant  Authenticator: EAPOL-Key(1,1,0,0,P,0,0,0,MIC,{}).

**12.7.6.3 4-way handshake message 2**

*At P 28.39 Modify 12.7.6.3 as shown below:*

* Additionally, contains an OCI KDE when dot11RSNAOperatingChannelValidationActivated is true on the Authenticator.
* Additionally, may include a Device ID KDE
* Additionally, may include an RRCM KDE
* The RSNXE that the Authenticator sent in its (Re)Association Request frame, if this element is present in the (Re) Association Request frame that the Authenticator sent.

Key Information:

Key Descriptor Version = 1 (ARC4 encryption with HMAC-MD5) or 2 (NIST AES key wrap with HMAC-SHA-1-128) or 3 (NIST AES key wrap with AES-128-CMAC), in all other cases 0 – same as message 1

Key Type = 1 (Pairwise) – same as message 1

Reserved = 0

Install = 0

Key Ack = 0

Key MIC = 0 when using an AEAD cipher or 1 otherwise

Secure = 0 – same as message 1

Error = 0 – same as message 1

Request = 0 – same as message 1

Encrypted Key Data = 1 when using an AEAD cipher or when RRCM KDE is included, or 0 otherwise

Reserved = 0 – unused by this protocol version

* Key Data =
* — Additionally, may include a Device ID KDE.

— Additionally, may include RRCM KDE.

* 4-way handshake message 3

*At P 28 Modify 12.7.6.4 as shown below:*

* Additionally, contains an OCI KDE when dot11RSNAOperatingChannelValidationActivated is true on the Authenticator.
* Additionally, may include a Device ID KDE
* Additionally, may include a MAAD KDE.
* The RSNXE that the Authenticator sent in its Beacon or Probe Response frame, if this element is present in the Beacon or Probe Response frame that the Authenticator sent.

***Add the following changes relevant to the use of KDK ((Proposed text modifications are based on Draft 802.11REVme\_D1.3))***

**a. (P342,line 1) 4.10.3.2 AKM operations with AS**

— If WUR frame protection is negotiated or RRCM generation is negotiated , derive a fresh WTK from the KDK

**b. (P3173,line30) under 12.6.1.1.6 PTKSA**

PTK(11ba), where the PTK includes the KDK when WUR frame protection is negotiated or RRCM is generated.

**c. (P3199,Line 64) under 12.7.1.1 General**

a) Pairwise key hierarchy, to protect individually addressed traffic(11ba), where the PTK includes a KDK if WUR frame protection is negotiated or RRCM generation is negotiated and excludes the KDK otherwise.

**d. (P3201, Line 50) under 12.7.1.3 Pairwise key hierarchy**

The PTK is partitioned into KCK, KEK, (11ba)a temporal key, and a KDK if WUR frame protection is negotiated or RRCM generation is negotiated ;otherwise the PTK is partitioned into KCK, KEK, and a temporal key. The temporal key is used by the MAC to protect individually addressed communication between the Authenticator’s and Supplicant’s respective STAs. If WUR frame protection is negotiated, the KDK is used to derive a WTK, which is used by the MAC of the WUR AP to protect and by the MAC of the WUR non-AP STA to validate individually addressed WUR Wake-up frames. PTKs are used between a single Supplicant and a single Authenticator. If RRCM generation is negotiated, the KDK is used to derive a RRMK, which is used to generate a batch of RMAs that are carried by the non-AP STA and identified by the AP.

**e. (P3202, Line 59) under 12.7.1.3 Pairwise key hierarchy**

where (11ba)Length = KCK\_bits + KEK\_bits + TK\_bits + KDK\_bits, if WUR frame protection is being negotiated or RRCM generation is being negotiated ;

**f.(P3203, Line 4) under 12.7.1.3 Pairwise key hierarchy**

(11ba)If WUR frame protection is being negotiated or RRCM generation is being negotiated, the KDK shall be computed as the next

KDK\_bits bits of the PTK:

KDK = L(PTK, KCK\_bits+KEK\_bits+TK\_bits, KDK\_bits)

Otherwise, the KDK is not derived

**g. (P3203,Line 32) under 12.7.1.3 Pairwise key hierarchy**

11ba)If WUR frame protection is negotiated, the WTK shall be derived from the KDK using the KDF

defined in 12.7.1.6.2:

WTK = KDF-Hash-Length(KDK, “WUR Temporal Key”, Min(AA,SPA) || Max(AA,SPA) ||

Min(ANonce,SNonce) || Max(ANonce,SNonce)

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 9-188 (AKM suite

selectors)).

— Length is the total number of bits to derive, i.e., number of bits of the WTK, and is equal to 128.

If RRCM is negotiated, the RRCMK shall be derived from the KDK using the KDF defined in 12.7.1.6.2. see subclause 12.2.12.2 **RMA and Key Generation**

**h. (P3211,Line 24) under 12.7.1.6.4 PMK-R1**

1ba)When WUR frame protection is negotiated or RRCM generation is negotiated, each PTK has six component keys, KCK, KEK, a

temporal key, KCK2, KEK2, and a KDK derived as follows:

(11ba)The KCK, KEK, temporal key, KCK2, and KEK2 shall be computed in the same way as when WUR frame protection is not negotiated.

(11ba)The KDK shall be computed as the next KDK\_bits bits of the PTK:

KDK = L(PTK, KCK\_bits+KEK\_bits+TK\_bits+KCK2\_bits+KEK2\_bits, KDK\_bits)

(11ba)The value of KDK\_bits is equal to the value of PMK\_bits (see 12.7.1.3 (Pairwise key hierarchy)).

**i. (insert the following change after the referenced baseline context in P3211,line 38) under 12.7.1.6.4 PMK-R1**

(11ba)If WUR frame protection is negotiated, the WTK shall be derived from the KDK using the KDF

defined in 12.7.1.6.2 (Key derivation function (KDF))):

WTK = KDF-Hash-Length(KDK, “WUR Temporal Key”, SNonce || ANonce || BSSID ||

STA-ADDR)

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 9-188 (AKM suite

selectors)).

— Length is the total number of bits to derive, i.e., number of bits of the WTK, and is equal to 128.

(11ba)The WTK is used to protect individually addressed WUR Wake-up frames, as defined in 29.10 (WUR

frame protection).

If RRCM is negotiated, the RRCMK shall be derived from the KDK using the KDF defined in 12.7.1.6.2. see subclause 12.2.12.2 **RMA and Key Generation**

**j. (P3226, line 42) under 12.7.6.2 4-way handshake message 1**

b) Derives PTK(11ba), the derived PTK including the Key derivation key (KDK) if WUR frame protection is being negotiated or RRCM generation is being negotiated .

**k. (P3269, line 54) under 12.11.2.5.3 PTKSA Key derivation with FILS authentication**

When the negotiated AKM is 00-0F-AC:16,FILS-FT is 256 bits; when the negotiated AKM is 00-0F-AC:17, FILS-FT is 384 bits; otherwise, FILS-FT is

not derived(11ba); when WUR frame protection is negotiated or RRCM generation is negotiated, the length of KDK is equal to the value of PMK\_bits (see 12.7.1.3 (Pairwise key hierarchy)); otherwise, the KDK is not derived.

**m. (P3270,line7) under 12.11.2.5.3 PTKSA Key derivation with FILS authentication**

11ba)When WUR frame protection is negotiated or RRCM generation is negotiated while doing FT initial mobility domain association using

FILS authentication,

KDK = L(PTK(#1778), ICK\_bits + KEK\_bits + TK\_bits + FILS-FT\_bits, KDK\_bits)

(11ba)When WUR frame protection is negotiated while not doing FT initial mobility domain association

using FILS authentication,

KDK = L(PTK(#1778), ICK\_bits + KEK\_bits + TK\_bits, KDK\_bits)

**n. (insert the following change after the referenced baseline context P3270,line 46) under 12.11.2.5.3 PTKSA Key derivation with FILS authentication**

11ba)If WUR frame protection is negotiated, the WTK shall be derived from the KDK using the KDF

defined in 12.7.1.6.2 (Key derivation function (KDF)):

WTK = KDF-Hash-Length(KDK, “WUR Temporal Key”, SPA || AA || SNonce || ANonce [ ||DHss ])

If RRCM is negotiated, the RRCMK shall be derived from the KDK using the KDF defined in 12.7.1.6.2. see subclause 12.2.12.2 **RMA and Key Generation**