Project	**IEEE 802.16 Broadband Wireless Access Working Group <http://ieee802.org/16>**
Title	**Suggested Modifications for AK Caching**
Date Submitted	**September 22, 2006**

Source(s)	Joe Schumacher, Steve Upp, Walter Goulet Motorola	Voice: +1 847 632 5978 [mailto:j.schumacher@motorola.com]
	Simon Mizikovsky, Robert Rance, Dan Gal, Peretz Feder Lucent Technologies	Voice: +1 973 386 6348 [mailto:smizikovsky@lucent.com]
	Phillip Barber Huawei, Inc	Voice: +1 972 365 6314 [mailto: pbarber@huawei.com]

Re:	802.16 Corrigendum 2
Abstract	Modifications are proposed that support the proposed AK Caching Solution without discarding the legacy CMAC calculation mode.
Purpose	Adoption
Notice	This document has been prepared to assist IEEE 802.16. It is offered as a basis for discussion and is not binding on the contributing individual(s) or organization(s). The material in this document is subject to change in form and content after further study. The contributor(s) reserve(s) the right to add, amend or withdraw material contained herein.
Release	The contributor grants a free, irrevocable license to the IEEE to incorporate material contained in this contribution, and any modifications thereof, in the creation of an IEEE Standards publication; to copyright in the IEEE's name any IEEE Standards publication even though it may include portions of this contribution; and at the IEEE's sole discretion to permit others to reproduce in whole or in part the resulting IEEE Standards publication. The contributor also acknowledges and accepts that this contribution may be made public by IEEE 802.16.
Patent Policy and Procedures	The contributor is familiar with the IEEE 802.16 Patent Policy and Procedures <http://ieee802.org/16/ipr/patents/policy.html>, including the statement "IEEE standards may include the known use of patent(s), including patent applications, provided the IEEE receives assurance from the patent holder or applicant with respect to patents essential for compliance with both mandatory and optional portions of the standard." Early disclosure to the Working Group of patent information that might be relevant to the standard is essential to reduce the possibility for delays in the development process and increase the likelihood that the draft publication will be approved for publication. Please notify the Chair <mailto:chair@wirelessman.org> as early as possible, in written or electronic form, if patented technology (or technology under patent application) might be incorporated into a draft standard being developed within the IEEE 802.16 Working Group. The Chair will disclose

this notification via the IEEE 802.16 web site <http://ieee802.org/16/ipr/patents/notices>.

AK Caching Solution

Joe Schumacher, Steve Upp, Walter Goulet (Motorola)
Simon Mizikovsky, Robert Rance, Dan Gal, Peretz Feder (Lucent Technologies, Inc.)
Philip Barber (Huawei, Inc.)

Problem Definition

Caching Issues

The secure message authentication method for the current 802.16e-2005 specification for initial entry, reentry, handover, secure location update, and diversity set entry/reentry is broken in that the current PKMv2 CMAC message authentication security method is unreasonably restrictive in its implementation and performance. The current method requires caching on the SS/MS, BS and the network Authenticator of AK context during the life of the PMK in order to protect from replay attacks. In anything but the most limited of systems, those comprising only a handful of base stations and a handful of mobiles, the memory requirements for retention of cached AK context become large. The memory requirements are large, but not necessarily untenable, for the SS/MS and BS. However, the most affected/encumbered component is the network Authenticator--the entity on the network responsible for key administration and distribution. This effect on the network Authenticator was not considered in the secure message authentication design in 802.16e-2005, and is not at all common to standard designs for such devices. As systems grow, these unrealistic limits--the memory requirements of mobiles, base stations, and most significantly Authenticators--become excessively large. Artificially limiting the scope of the Authenticator to cover only a few BSs, while mitigating the problem, would result in substantially more frequent re-authentications and overall poor handover performance. A relatively simple addition of a TLV and change to CMAC digest tuple calculation removes this onerous memory burden, preserves protection from replay attacks, and permits the employment of more common design network Authenticators.

However, for systems in which network entry and handover are infrequent or rare events, the legacy CMAC key generation algorithm is serviceable and should be retained.

Remedy

To solve the AK Caching problem while conforming to the above security model, we propose to freshly generate new CMAC keys after each network entry and handover. The method of generating these keys is defined. The new TLV is defined for the RNG-REQ message that contains the mobile-maintained counter, called the CMAC_KEY_COUNT, used to generate the fresh value of CMAC keys for every access.

Text is proposed that supports both the proposed and legacy CMAC key generation algorithms. The legacy CMAC key generation algorithm is designated CMAC-0 and the proposed CMAC key generation algorithm is designated CMAC.

Proposed Text Changes

Change 1. Modify generation of the CMAC_KEY_*

[Please modify the text as follows in the section 7.2.2.2.9, page 280 of 802.16e-2005]

7.2.2.2.9 Message authentication keys (HMAC/CMAC) and KEK derivation

MAC (message authentication code) keys are used to sign management messages in order to validate the authenticity of these messages. The MAC to be used is negotiated at SS Basic Capabilities negotiation.

There is a different key for UL and DL messages. Also, a different message authentication key is generated for a multicast message (this is DL direction only) and for a unicast message.

In general, the message authentication keys used to generate the CMAC value and the HMAC-Digest are derived from the AK.

CMAC Keys are computed differently based on the CMAC algorithm version negotiated via the MAC Mode TLV (see Section 11.8.4.3)

[Insert new section 7.2.2.2.9.1 as indicated]

7.2.2.2.9.1 CMAC-0 key derivation

If bit #1 of the MAC Mode TLV is set to '1' (see Section 11.8.4.3), the keys used for CMAC key and for KEK are derived as follows:

CMAC_KEY_U | CMAC_KEY_D | KEK ⇐ Dot16KDF(AK, SS MAC Address | BSID | "CMAC_KEYS+KEK", 384)

CMAC_KEY_GD ⇐ Dot16KDF(GKEK, "GROUP CMAC KEY",128) (Used for multicast MAC message such as a PKMv2 Group-Key-Update-Command message)

[Insert new section 7.2.2.2.9.2 as indicated]

7.2.2.2.9.2 CMAC key derivation

If bit #5 of the MAC Mode TLV is set to '1' (see Section 11.8.4.3) the keys used for CMAC key and for KEK are as follows:

CMAC_PREKEY_U | CMAC_PREKEY_D | KEK ⇐ Dot16KDF(AK, SS MAC Address | BSID | "CMAC_KEYS+KEK", 384)

CMAC_KEY_GD ⇐ Dot16KDF(GKEK, “GROUP CMAC KEY”,128) (Used for multicast MAC message such as a PKMv2 Group-Key-Update-Command message)

CMAC_KEY_U ⇐ $AES_{CMAC\_PREKEY\_U}$(CMAC_KEY_COUNT)
CMAC_KEY_D ⇐ $AES_{CMAC\_PREKEY\_D}$(CMAC_KEY_COUNT)

Specifically, the preprocessed value of CMAC_PREKEY_* is treated as the Cipher Key of the Advanced Encryption Standard (AES) algorithm AES128 (FIPS197). The CMAC_KEY_COUNT is treated as the Input Block Plain Text of this algorithm. The AES128 algorithm is executed once. The Output Block Cipher Text of this algorithm is treated as the resulting CMAC_KEY_*. When CMAC_KEY_COUNT is used as an input of AES128 algorithm, 112 zero bits are prepadded before the 16 bit CMAC_KEY_COUNT where the CMAC_KEY_COUNT is regarded as most-significant-bit first order. The AES input is also defined as most-significant-bit first order.

[Insert new section 7.2.2.2.9.3 as indicated]

7.2.2.2.9.3 Further message authentication key derivation

~~CMAC_KEY_GD ⇐ Dot16KDF(GKEK, “GROUP CMAC KEY”,128) (Used for multicast MAC message such as a PKMv2 Group-Key-Update-Command message)~~

The keys used for HMAC key and for KEK are as follows:

HMAC_KEY_U | HMAC_KEY_D | KEK ⇐ Dot16KDF(AK, SS MAC Address | BSID | “HMAC_KEYS+KEK”, 448)
HMAC_KEY_GD ⇐ Dot16KDF(GKEK, “GROUP HMAC KEY”, 160) (Used for multicast MAC message such as a PKMv2 Group-Key-Update-Command message)

Exceptionally, the message authentication keys for the HMAC/CMAC Digest included in a PKMv2 Authenticated-EAP-Transfer message are derived from the EIK instead of the AK

The keys used for CMAC key and for KEK are as follows:

CMAC_KEY_U | CMAC_KEY_D ⇐ Dot16KDF(EIK, SS MAC Address | BSID | “CMAC_KEYS”, 256)

The keys used for HMAC key and for KEK are as follows:
HMAC_KEY_U | HMAC_KEY_D ⇐ Dot16KDF(EIK, SS MAC Address | BSID | “HMAC_KEYS”, 320)

Change 2: Modify text in description of AK context, modify text in AK Lifetime usage cell, and add a parameter called CMAC_KEY_COUNT to the table.

[Please modify the text as follows in the section 7.2.2.4.1, page 287 of 802.16e-2005]

7.2.2.4.1 AK context

The PMK key has two phases of lifetime: the first begins at PMK creation and the second begins after validation by the 3-way handshake.
The phases ensure that when the PMK is created it will be defined with the PMK or PAK pre-handshake lifetime and after successful 3-way handshake, this lifetime may be enlarged using the PMK lifetime TLV within the 3-way handshake.

For CMAC-0, HMAC and short-HMAC modes, if the cached AK and associated context is lost by either the BS or SS, no new AKs can be derived from this PMK on handover.

Cached AKs that were derived from the PMK can continue to be used in HO.

Reauthentication is required to obtain a new PMK so as to derive new AKs.

Table 133a AK Context in PKMv2

Parameter	Size (bits)	Usage
AK	160	The authorization key, calculated as defined in 7.2.2.2.3.
AKID	64	AKID = Dot16KDF(AK, AK SN|SS MAC Address|BSID|"AK", 64) The AK_SN in the Dot16KDF function is an 8-bit number which consists of leading 4 zero bits and appending 4-bit AK_SN in MSB first order.
AK Sequence Number	4	Sequence number of root keys (PAK and PMK) for the AK. This value is the most significant 2-bit of PAK sequence number concatenated with the least significant 2-bit of PMK sequence number. If AK = f (PAK and PMK), then AK SN = PAK SN + PMK SN If AK = f (PAK), then AK SN = PAK SN

		If AK = f (PMK), then AK SN = PMK SN
AK Lifetime	_	This is the time this key is valid; it is calculated as AK lifetime = MIN(PAK lifetime, PMK lifetime)—when this expires, re-authentication is needed.
PMK Sequence Number	4	The sequence number of the PMK from which this AK is derived.
HMAC/CMAC_KEY_U	160/128	The key which is used for signing UL management messages.
HMAC/CMAC_PN_U	32	Used to avoid UL replay attack on the management connection – when this expires, re-authentication is needed. If CMAC-0 is negotiated, the initial value of CMAC_PN_U is zero. The CMAC_PN_U is cached by the MS and BS for the life of the AK. If CMAC is negotiated, the initial value of CMAC_PN_U is zero and the value of CMAC_PN_U is reset to zero whenever CMAC_KEY_COUNT is increased.
HMAC/CMAC_KEY_D	160/128	The key which is used for signing DL management messages.
HMAC/CMAC_PN_D	32	Used to avoid DL reply attack on the management connection – when this expires, re-authentication is needed. If CMAC-0 is negotiated, the initial value of CMAC_PN_D is zero. The CMAC_PN_D is cached by the MS and BS for the life of the AK. If CMAC is negotiated, the initial value of CMAC_PN_U is zero and the value of CMAC_PN_D is reset to zero whenever CMAC_KEY_COUNT is increased.
KEK	160	Used to encrypt transport keys from the BS to the SS.
EIK	160	EAP Integrity Key for authenticating Authenticated EAP message.
CMAC_KEY_COUNT	16	Value of the Entry Counter that is used to guarantee freshness of computed CMAC_KEY_* with every entry and provide replay protection.

Change 3: Modify text in 7.5.4.4.1 “Calculation of CMAC Value”

[Please modify the text as follows in section 7.5.4.4.1, page 305 of 802.16e-2005]

7.5.4.4.1 Calculation of CMAC Value

The calculation of the keyed hash value contained in the CMAC-Digest attribute and the CMAC Tuple shall use the CMAC Algorithm with AES. The downlink authentication key CMAC_KEY_D shall be used for authenticating messages in the downlink direction. The uplink authentication key CMAC_KEY_U shall be used for authenticating messages in the uplink direction. Uplink and downlink message authentication keys are derived from the AK (see ~~7.5.4 below~~ 7.2.2.2.9 for details).
For authentication multicast messages (in the DL only) a CMAC_KEY_GD shall be used (one for each group), group authentication key is derived from GKEK
The CMAC-Digest and CMAC Tuple attributes shall be only applicable to the PKM version 2. In the PKM version 2 protocol, the AKID ~~CMAC key sequence number~~ used in the computation of the CMAC value ~~tuple~~ shall be the 64 bit AKID ~~equal to the 4-bit AK sequence number~~ of the AK from which the CMAC_KEY_x was derived. See 6.3.2.3.9.18 for the SA-TEK-Challenge message attributes in which the mapping between the AK Sequence number and the AKID is communicated and see 7.2.2.4.1 for a description of the AK context that contains the AK and AKID.
The CMAC Packet Number Counter (CMAC_PN_*) is a 4-byte sequential counter that is incremented in the context of UL messages by the SS, and in the context of DL messages by the BS,. The BS will also maintain a separate CMAC_PN_* for multicast packets per each GSA and increment that counter in the context of each multicast packet from the group. For MAC messages that have no CID e.g., RNG-REQ message, the CMAC_PN_* context will be the same as used on the basic CID. If basic CID is unknown (e.g., in network reentry situation) then CID 0 should be used.
The CMAC Packet Number Counter, CMAC_PN_*, is part of the CMAC security context and must be unique for each MAC management message with the CMAC tuple or digest. Any tuple value of {CMAC_PN_*, CMAC_KEY_* ~~AK~~} shall not be used more than once. The re-authentication process should be initiated (by BS or SS) to establish a new AK before the CMAC_PN_* reaches the end of its number space.
The digest shall be calculated over a field consisting of the AKID ~~CMAC key sequence number~~ followed by the CMAC Packet Number Counter, expressed as an unsigned 32-bit number, followed by the 16-bit Connection ID on which the message is sent, followed by 16-bit of zero padding (for the header to be aligned with AES block size) and followed by the entire MAC management message with the exception of the CMAC-TLV.
The least significant bits of the digest shall be truncated to yield a 64-bit length digest. The CMAC key sequence number shall be equal to the 4-bit AK sequence number of the AK from which the CMAC_KEY_x was derived.
i.e.,:
CMAC value <= Truncate64 (CMAC (CMAC_KEY_*, AKID ~~CMAC key sequence number~~ | CMAC_PN | CID |16-bit zero padding | MAC_Management_Message))

If the digest is included in an MPDU that has no CID, e.g., A RNG-REQ message, the CID used shall take the value of the basic CID. If basic CID is unknown (e.g., in network reentry situation) then CID 0 should be used.

Change 4: Modify text in 7.5.4.6.1 “Dot16KDF for PKMv2”

[Please modify the text in section 7.5.4.6.1, page 308 of 802.16e-2005]

The Dot16KDF algorithm is a CTR mode construction that may be used to derive an arbitrary amount of keying material from source keying material.

In the case that the HMAC/CMAC setting in the MAC (Message Authentication Code) mode is set to ~~CMAC~~ either CMAC-0 or CMAC, the algorithm is defined as:

```
Dot16KDF(key, astring, keylength)
{
        result = null;
        Kin = Truncate (key, 128);
        for (i = 0;i <= int((keylength-1)/128); i++) {
                result = result | CMAC(Kin, i | astring | keylength);
        }
        return Truncate (result, keylength);
}
```

In the case that the HMAC/CMAC setting in the authentication policy bits is set to HMAC, the algorithm is defined as:

```
Dot16KDF(key, astring, keylength)
{
        result = null;
        Kin = Truncate (key, 160);
        For (i=0; i <= int( (keylength-1)/160 ); i++) {
                result = result | SHA-1( i| astring | keylength | Kin);
        }
        return Truncate (result, keylength);
}
```

Change 5: Insert new row “CMAC_KEY_COUNT” into the RNG-REQ table in 11.5

[Please add the following row at the end of the table 364 in section 11.5, page 678 of 802.16e-2005]

Name	Type (1 byte)	Length	Value (Variable-length)
...	...	...	...
CMAC Key_Count	13	2	CMAC_KEY_COUNT A 16-bit counter that identifies the number of times the CMAC_KEY_* is computed by the MS and BS while the same value of PMK is active (e.g. during the time from one authentication or re-authentication to the next). The value of this counter is used for the generating fresh CMAC_KEY_D and CMAC_KEY_U keys, and therefore prevents replays of the MAC management messages. A message received, that contains an CMAC Tuple, shall not be considered authentic if the length field of the tuple is incorrect, or if the locally computed value of the digest does not match the digest in the message. NOTE: It would be appropriate for a MIB to increment an error count on receipt of a non authentic message, so that management can detect an active attack.

Change 6: Add CMAC_KEY_COUNT to RNG-REQ message.

[Please insert the red text as follows in the line 28 of the section 6.3.2.3.5, page 52 of 802.16e-2005]

Power_Saving_Class_Parameters

Compound TLV to specify Power Saving Class operation.

The following- parameters may be included in the RNG-REQ message when the MS is attempting to perform network re-entry or handover and the MS has a valid HMAC/CMAC Tuple necessary to expedite security authentication.

CMAC_KEY_Count

The value of this counter is reset to zero by both MS and BS upon successful completion of the PKMv2 EAP based authentication or re-authentication, and is incremented for each MS access to the BS. When the MS decides either to reenter the network, handover to a target BS, or perform a secure Location Update, it enters its CMAC_Key_Lock state as part of this process. While in this state, its CMAC_KEY_COUNT cannot be changed. Any RNG-REQ messages sent to other potential target BSs while in the CMAC_Key_Lock state will use the same CMAC_KEY_COUNT. The CMAC_PN_* counts corresponding to each potential target BS that was contacted with an RNG-REQ message will be cached by the MS while it is in the Lock State.
When the MS decides that it is either connected to the target BS, or declines handover and remains connected to its current serving BS, it enters its CMAC_Key_Unlock state. Normally, the value of CMAC_KEY_COUNT will be maintained in synchronicity by MS and BS. If the target BS receives and authenticates an RNG-REQ message containing a CMAC_KEY_COUNT higher than its own, it shall adopt the received count. Other details on re-synchronization of this counter in the BS and network is outside the scope of this document.
Whenever the CMAC Tuple is included in the RNG-REQ message during the entry, re-entry, secure Location Update, or HO, the CMAC_KEY_Count TLV shall be included.

HMAC/CMAC Tuple (see 11.1.2)
The HMAC/CMAC Tuple shall be the last attribute in the message.

Change 6: Modify CMAC Key derivation figure.

[Please substitute the following diagram for Figure 130o, page 283 of 802.16e-2005]

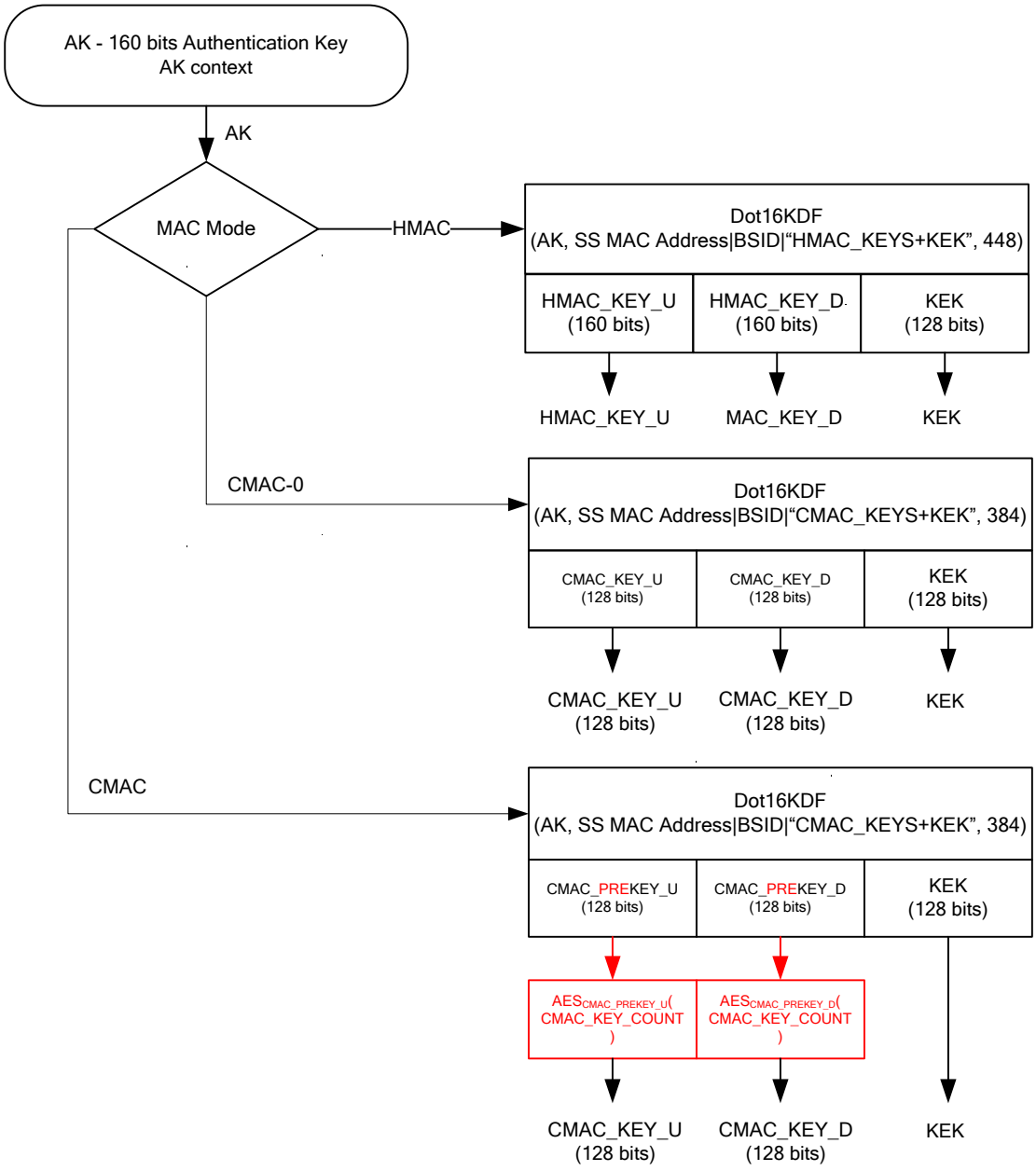

Figure 130o—HMAC/CMAC/KEK derivation from AK

Change 7: Modify MAC Mode TLV.

[Please modify the text as follows for the MAC Mode TLV in Section 11.8.4.3, pages 711-712 of 802.16e-2005]

Type	Length	Value
25.3	1	Bit# 0: HMAC Bit# 1: CMAC**-0** Bit# 2: 64-bit short-HMAC$_a$ Bit# 3: 80-bit short-HMAC$_a$ Bit# 4: 96-bit short-HMAC$_a$ **Bit #5: CMAC** Bit# ~~5~~**6**-7: Reserved. Set to 0