Project	**IEEE 802.16 Broadband Wireless Access Working Group <http://ieee802.org/16>**
Title	**Key Hierarchy Text for PKMv2**
Date Submitted	**2004-6-21**
Source(s)	David Johnston Intel Corporation 2111 NE 25th Ave. Hillsboro 97124 Voice: +1 (503) 264-3855 [mailto:dj.johnston@intel.com]
Re:	IEEE 802.16e Security Adhoc
Abstract	Proposal to introduce a key hierarchy to PKMv2
Purpose	To create an well defined hierarchy for the storage and derivation of keys in PKMv2
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>.

PKMv2 Key Hierarchy

David Johnston, Intel
JunHyuk Song, Samsung
Seokheon Cho, ETRI
Donnie Lee, SKT
Jesse Walker, Intel

PKM version 1 defines keys and their relationships, but a clear definition of the total key hierarchy is not present. Neither is the context in which the AK is stored. There are vulnerabilities associated with this.

This proposal creates a well defined set of security contexts and a clear key hierarchy for use in PKMv2. The vulnerabilities identified in PKMv1 are avoided.

Two keying models are assumed. The first is a model where the master key at each base station is different between different base stations. A full secure network entry is required to establish keying material when a mobile station hands over.

The second model is where pre-authentication is used. The AAA server and the MSS share a master key (The MK). The PMK is derived from the master key, and the identities of the BS and MSS. Thus there is a different AK per {BS,SS} tuple. The AAA server populates BSs with the per {BS,SS} AK and this key will be known to both the SS and the BS, so an SS does not need to enter into a key establishment procedure with the target BS during a handover.

There is a PAK that may be derived from a certificated RSA exchange.

There is an AK, this is equivalent to the AK in the existing PKMv1 protocol. The AK is derived from the PMK and/or the PAK, whichever is available.

A generic PRF is defined to be used to derive keys.

```
Dot16KDF(key, astring, keylength) =
{
        result = null;
        For (i = 0;i < int(keylength/128); i++)
        {
                result <= result | OMAC(key, i | astring | keylength);
        }
```

```
    return result;
}
```

So:

PAK is generated randomly in the BS and transmitted to the SS, encrypted with RSA during the PKMv2 authorization exchange.

EEK (EAP Encryption Key) = Dot16KDF(PAK, “EPK”, 128)
EIK (EAP Integrity Key) = Dot16PRF(PAK, “EIK”, 128)

MK is generated in the AAA server and the SS as a result of an EAP based authentication exchange.

PMK[0..127] = Dot16PRF(MK, 0x00 | BSID | SSID | 256)
PMK[128,255] = Dot16PRF(MK, 0x01 | BSID | SSID | 256)

```
If (RSA has been used yielding a PAK and EAP has been used, yielding a PMK) then
    AK <= Dot16KDF(PMK, PAK | 0x80);
Elsif (EAP has been used, yielding a PMK)
    AK <= PMK
Elsif only RSA has been used then
    AK <= PAK
Else
    No security mode is selected
End if
```

HMAC_KEY_U <= Dot16PRF(AK, “HMAK_KEY_U” | 0x80)
HMAC_KEY_D <= Dot16PRF(AK, “HMAC_KEY_D” | 0x80)

OMAC_KEY_U <= Dot16PRF(AK, “OMAC_KEY_U” | 0x80)
OMAC_KEY_D <= Dot16PRF(AK, “OMAC_KEY_D” | 0x80)

KEK <= Dot16PRF(AK, “KEK” | 0x080)

TEK is encrypted with KEK using AES_KEY_WRAP with a KEK and transferred between BS and SS

GKEKEK (GKEK Encryption Key) <= Dot16PRF(AK, “GKEKEK” | 0x80)
GKEKIK (GKEK Integrity Key) <= Dot16PRF(AK, “GKEKIK” | 0x80)

GKEK is encrypted with GKEKEK and integrity checked with GKEKIK.

In the event that the MBS proposal is accepted…

MAK is encrypted with the MAKEK, integrity checked with the MAKIK transferred to the MBS GSA. The keys within the MBS GSA are the same as for a GSA, however the MAK is used in place of the AK.

MKEK and MTEK rules as per unicast, but MAK and MTEK are common within group.
MBS_traffic_key = Dot16PRF(MAK, MTEK, 128)

For MBS services, the GSAID is caused to be the same amongst group members, by the BS enforcing it.

Editor Instructions:
[In the appropriate PKMv2 part of of clause 7, insert:]

7.x.x.x Key Hierarchy

The PKMv2 key hierarchy defines what keys are present in the system how keys they are generated.
Since there are two authentication schemes, one based on RSA and one based on EAP, there are two primary sources of keying material.

Keying material is held within associations. There are four types of association: The Authorized Association (AA) that is the top of the security hierarchy, security associations (SA) that maintain keying material for unicast connections, group security associations (GSA) that hold keying material for multicast groups.

[If the Samsung MBS proposal is accepted, insert the following line here:]
and MBSGSAs which hold keying material for MBS services.

[Insert]

7.x.x.x Authorized Association

The AA (Authorized Association) contains the primary keying material of the PKMv2 protocol. The contents are:

The AAID, a 16 bit identifier for the AA. This AAID shall be unique within a BS.
The Primary AK (PAK). A 128 bit key yielded from the authorization exchange. The PAK is only present if the certificated RSA exchange took place, as a result of the authorization policy negotiation.

The Master Key (MK). A 128 bit key yielded from the successful completion of an EAP method during the EAP exchange. The MK is only present of the EAP exchange took place, as a result of the authorization policy negotiation.
The Primary Master Key (PMK). A 128 bit key derived from the MK and the identities of both endpoints sharing the key.
The AK. A 128 bit authorization key.
The AK Key lifetime
The EAP Encryption Key (EEK), a 128 bit key that may be used to protect EAP traffic.
The EAP Integrity Key (EIK), a 128 bit key that may be used to integrity protect EAP traffic.
The uplink OMAC key (OMAC_KEY_U), a 128 bit key used as the keying material for the generation of uplink OMAC digests.
The downlink OMAC key (OMAC_KEY_D), a 128 bit key used as the keying material for the generation of downlink OMAC digests.
The Group Key Encryption Key Encryption Key (GKEKEK), a 128 bit key used to encrypt keying material transferred into a GSA.
The Group Key Encryption Key Integrity Key (GKEKIK), a 128 bit key used to integrity protect keying material transferred into a GSA.
The SAID list. A list of SAs secured under the AA.
The GSA list. A list of GSAs secured under the AA.
The MBSGSA list. A list of MBSGSAs secured under the AA.

7.x.x.x Security Associations

A security association contains keying material that is used to protect unicast connections. The contents of an SA are:

The SAID, a 16 bit identifier for the SA. The SAID shall be unique within a BS.
The KEK, a 128 bit key encryption key, derived from the AK.
The KIK, a 128 key integrity key, derived from the AK.
TEK0 and TEK1, 128 bit traffic encryption keys, generated within the BS and transferred from the BS to the SS using a secure key exchange.
The TEK Lifetimes TEK0 and TEK1, a key aging lifetime value.
PN0 and PN1, 32 bit packet numbers for use by the link cipher
RxPN0 and RxPN1, 32 bit receive sequence counter, for use by the link cipher.

7.x.x.x Group Security Association

The Group Security Association (GSA) contains keying material used to secure multicast groups. These are defined separately from SAs since GSA offer a lower security bound that unicast security associations, since

keying material is shared between all members of the group, allowing any member of the group to forge traffic as if it came from any other member of the group.

The contents of a GSA are:

The Group Key Encryption Key (GKEK). Serves the same function as an SA KEK but for a GSA
The Group Traffic Encryption Key (GTEK). Served the same function as an SA TEK but for a GSA.

[In the event that the Samsung MBS proposal is accepted, insert:]

7.x.x.x MBS Group Security Association

The primary keying material in the MBS Group Security Association is the MAK. This serves the same function as the AK in the Authorized Association, however the MAK is provisioned by an external entity, such as an MBS server. The MAK may be common between members of an MBS group.

The contents of an MBSGSA are:

The MAK, a 128 bit MBS AK, serves the same function as the AK but local to the MBSGSA.
The MKEK, a 128 bit MBS Key Encryption Key, used to encrypt keying material transferred from BS to SS.
The MKIK, as 128 bit MBS Key Integrity Key, used to integrity protect keying material transferred from BS to SS.
The MGTEK, a 128 bit MBS Group Traffic Encryption Key, used indirectly to protect MBS traffic. It is updated more frequently that the MAK.
The MTK, MBS Traffic Key, a 128 bit key used to protect MBS traffic, derived from the MAK and MGTEK.

7.x.x.x. Key Derivation

All PKMv2 key derivations are based on the Dot16PRF algorithm as defined in 7.x.x.x Dot16PRF.

EEK <= Dot16KDF(PAK, “EPK”, 128)
EIK <= Dot16PRF(PAK, “EIK”, 128)

MK is generated in the AAA server and the SS as a result of an EAP based authentication exchange.

PMK[0..127] = Dot16PRF(MK, 0x00 | BSID | SSID | 256)
PMK[128,255] = Dot16PRF(MK, 0x01 | BSID | SSID | 256)

If RSA has been used yielding a PAK and EAP has been used, yielding a PMK then the following AK construction shall be used:

AK <= Dot16KDF(PMK, PAK | 0x80);

Else if EAP has been used, yielding a PMK then the following AK contruction shall be used:

AK <= PMK

Else if only RSA has been used then

AK <= PAK

OMAC_KEY_U <= Dot16PRF(AK, "OMAC_KEY_U" | 0x80)
OMAC_KEY_D <= Dot16PRF(AK, "OMAC_KEY_D" | 0x80)

KEK <= Dot16PRF(AK, "KEK" | 0x080)

TEK is encrypted with KEK using AES_KEY_WRAP with a KEK and transferred between BS and SS
GKEKEK (GKEK Encryption Key) <= Dot16PRF(AK, "GKEKEK" | 0x80)
GKEKIK (GKEK Integrity Key) <= Dot16PRF(AK, "GKEKIK" | 0x80)
GKEK is encrypted with GKEKEK and integrity checked with GKEKIK.

[In the event that the MBS proposal is accepted…]

MAK is encrypted with the MAKEK, integrity checked with the MAKIK transferred to the MBS GSA. The keys within the MBS GSA are the same as for a GSA, however the MAK is used in place of the AK.

MTK <= Dot16PRF(MAK, MTEK, 128)

[Insert under the cryptographic algorithms section]

7.x.x.x.x Dot16PRF

The Dot16PRF algorithm is a CTR mode construction that may be used to derive an arbitrary amount of keying material from source keying material.

The algorithm is defines as:

```
Dot16KDF(key, astring, keylength) is
{
        result = null;
        For (i = 0;i < int(keylength/128); i++)
```

```
        {
                result <= result | OMAC(key, i | astring | keylength);
        }
        return result;
}
```

The key is a cryptographic key that is used by the underlying OMAC algorithm. ‘astring’ is an octet string used to alter the output of the algorithm. ‘keylength’ is used to prevent extension attacks.