IEEE P802.15
Wireless Personal Area Networks

Project	IEEE P802.15 Working Group for Wireless Personal Area Networks (WPANs)		
Title	**LB97 resolutions - Kivinen**		
Date Submitted	11 July, 2014		
Source	Tero Kivinen INSIDE Secure Eerikinkatu 28 FI-00180 Helsinki Finland	Voice: Fax: E-mail:	+358 20 500 7800 +358 20 500 7801 kivinen@iki.fi
Re:	LB97 resolutions		
Abstract	LB 97 resolutions to CIDs 1028, 1150, 1026, 1145, 1146, 1148, 1147, 1153, 1101, 1151, 1105, 1104, 1154, R41 and R42		
Purpose	LB97 resolutions		
Notice	This document has been prepared to assist the IEEE P802.15. 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 acknowledges and accepts that this contribution becomes the property of IEEE and may be made publicly available by P802.15.		

CID 1028

Robert Moskowitz	Verizon	313	9.2	16	Still need state machines for security flow	Provide state machine figures

Work In Progress, Outbound state machine done, see CID 1150 for it.

CID 1150

Tero Kivinen	INSIDE Secure	313	9.2.1	37	Swap steps c) and b). There is no point of checking length of the data expansion length if we are not protecting the frame. See CID 565.	Swap steps c) and b).

New version of the Outbound state machine:

The inputs to this procedure are the *frame to be secured* and the *SecurityLevel*, *KeyIdMode*, *KeySource*, and *KeyIndex* parameters from the originating primitive or automatic request PIB attributes. The outputs from this procedure are the *status* of the procedure and, if this *status* is SUCCESS, the *secured frame*.

The outgoing frame security procedure involves the following steps:

a) Do we need to secure the packet?

If the *SecurityLevel* parameter is zero, the procedure shall set the *secured frame* to be the *frame to be secured* and return with a *status* of SUCCESS.

b) Do we have security enabled?

If the *macSecurityEnabled* attribute is set to FALSE the procedure shall return with a *status* of UNSUPPORTED_SECURITY.

c) Fetch the KeyDescriptor.

The procedure shall obtain the *KeyDescriptor* using the KeyDescriptor lookup procedure as described in 9.2.2 with the *device addressing mode* set to DstAddrMode, the *device PAN ID* set to DstPANId, and the *device address* set to DstAddr. If that procedure fails, the procedure shall return with a *status* of UNAVAILABLE_KEY.

d) Fetch frame counter.

If not using TSCH mode, the procedure shall set the *frame counter* to the *macFrameCounter* attribute. If the *frame counter* has the value 0xffffffff, the procedure shall return with a *status* of COUNTER_ERROR.

e) Insert and fill auxiliary security header.

The procedure shall insert the auxiliary security header into the frame, with fields set as follows:

1) The Security Level field of the Security Control field shall be set to the *SecurityLevel* parameter.

2) The Key Identifier Mode field of the Security Control field shall be set to the *KeyIdMode* parameter.

3) If not using TSCH mode, the Frame Counter field shall be set to the *frame counter*.

4) If the *KeyIdMode* parameter is set to a value not equal to zero, the Key Source and Key Index fields of the Key Identifier field shall be set to the *KeySource* and *KeyIndex* parameters, respectively.

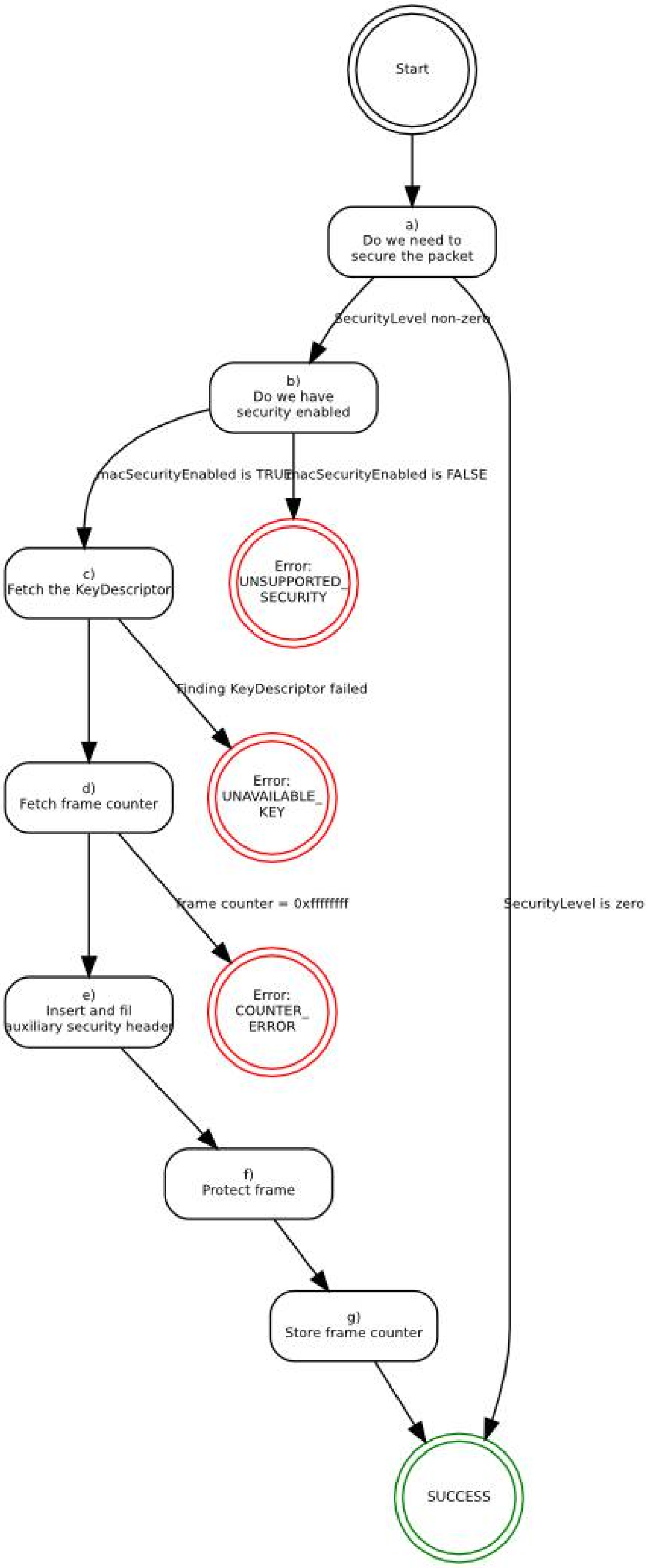
f) Protect frame.

For frames specified in the Table 146 the *Private Payload field* and *Open Payload field* shall be set as indicate there. For frames not specified in Table 146 the *Private Payload field* shall be set to contain all MAC payload fields, and the *Open Payload field* shall be empty. The procedure shall then use the *Private Payload field*, the *Open Payload field*, the *macExtendedAddress*, the *frame counter*, the *SecurityLevel* parameter, and the Key element of the *KeyDescriptor* to produce the *secured frame* according to the CCM* transformation process defined in 9.3.4.

g) Store frame counter

If not using TSCH mode, the procedure shall increment the *frame counter* by one and set the *macFrameCounter* attribute to the resulting value.

h) The procedure shall return with a *status* of SUCCESS.



CID 1026, 1145, 1146, 1148, 1147

Toyoyuki Kato	Anritsu Engineering Co.,Ltd,	314	9.2.1	17	"Table 146" is incorrect.	Correct it appropriately.
Tero Kivinen	INSIDE Secure	314	9.2.1	25	In Table 146 we say that in beacons following fields are not encrypted: superface specifications, gts info, pending address, and both header IEs and Payload IEs. Are Payload IEs really meant to be sent unencrypted?	Add Payload IEs to the private payload fields.
Tero Kivinen	INSIDE Secure	314	9.2.1	26	In Table 146 we say that in Data only the Data Payload field is encrypted, all other fields are unencrypted. This includes both header and Payload IEs. I would assume we want to make Payload IEs encrypted.	Add Payload IEs to the private payload fields.
Tero Kivinen	INSIDE Secure	314	9.2.1	31	In Table 146 we say that there is no Open Payload fields for the MAC Command frame with version number >= 2, but the private payload fields does not list command identifier. So we do not include Command Identifier in either column, so we do not know whether it is open or private field?	Add Command Identifier to the Private Payload field column.
Tero Kivinen	INSIDE Secure	314	9.2.1	33	In Table 146 we say that for Acknowledgement frames the full Information Elements field is encrypted, this includes both Header IE and Payload IE fields. Is this intended, or should we only include Payload IE here? If Header IE field is also included to be protected, then we most likely want to protect them also in other frame formats.	Clarify.

Change inbound and outbound processing rules to say that Table 146 only contains exceptions to the generic rule, and that generic rule is that Private Payload Field contains all MAC Payload fields, and Open Payload Field is empty. Change the Table 146 to contain:

Frame type	Private Payload Field	Open Payload Field
Beacon (Frame Version < 2)	Beacon Payload	All other fields in the MAC Payload
MAC Command (Frame Version < 2)	Content	Command Identifier

CID 1153, 1101, R41, R42

Tero Kivinen	INSIDE Secure	315	9.2.3	49	Both steps c) and steps e) will set “key identifier mode” / “KeyIdMode”, “key source”/”KeyIndex” and “key index”/”KeyIndex”. I think we need to do this only once.	Modify the step c) so it will copy the values out from the auxiliary security header, and remove the copying from step e), so all auxiliary security header processing is in one step. We need to check which names of those local variables needs to be used in the rest of the processing steps, and perhaps also change them to use separate typographical look.
Tero Kivinen	INSIDE Secure	316	9.2.3	50	CID 470 was not done: In step l) the step will check the frame counter value of 0xffffffff. With TSCH mode the frame counter is not used, instead of 5-octet absolute slot number ASN is used, this test is not needed. Actually the current draft accidently did that change for step i) not for step l), so move the text from step “i) … procedure shall determine whether the frame to be unsecured ...” to step “l) The procedure shall set frame counter …”.	Add ”If not using TSCH mode” in front of step l.
Tero Kivinen	INSIDE Secure	316	9.2.3	35	Steps i), j) and k) should be folded in to the step h. It is stupid to call subprocedure to fetch the SecurityLevelDescriptor and then check it here, as we could call SecurityLevelDescriptor validation procedure, that will get that SecurityLevelDescriptor and verify the packet is according to it, and then either return error (either UNAVAILABLE_SECURITY_LEVEL or IMPROVED_SECURITY_LEVEL) or SUCCESS, or pass forward.	As described in the Comment section.
Tero Kivinen	INSIDE Secure	316	9.2.3	49	Steps l and m should be combined.	Replace with “If not using TSCH mode, the procedure shall set frame counter to the Frame Counter field of the frame to be unsecured. If frame counter has the value 0xffffffff, or if the frame counter is less than the FrameCounter element of the DeviceDescriptor, the procedure shall return with a status of COUNTER_ERROR.”

The input to this procedure is the *frame to be unsecured*. The outputs from this procedure are the *status* of the procedure and, if this *status* is SUCCESS, the *unsecured frame*, the *security level*, the *key identifier mode*, the *key source*, and the *key index*.

All outputs of this procedure are assumed to be invalid unless and until explicitly set in this procedure.

The incoming frame security procedure involves the following steps:

a) Do we have secured frame?

If the Security Enabled field of the *frame to be unsecured* is set to zero, the procedure shall use procedure described in section 9.2.3b.

b) Legacy security?

If the Frame Version field of the *frame to be unsecured* is set to zero, the procedure shall return with a *status* of UNSUPPORTED_LEGACY.

c) Check for macSecurityEnabled

If the *macSecurityEnabled* attribute is set to FALSE, the procedure shall return *status* of UNSUPPORTED_SECURITY.

d) Parse Auxiliary Security Header

The procedure shall set the *security level* and the *key identifier mode* to the corresponding fields of the Security Control field of the auxiliary security header of the *frame to be unsecured*, and the *key source* and *key index* to the corresponding fields of the Key Identifier field of the auxiliary security header of the *frame to be unsecured*, if present. If the resulting *security level* is zero, the procedure shall return with a *status* of UNSUPPORTED_SECURITY

e) Get Source Address

The *device PAN ID* shall be set to the Source PAN Identifier field, if it is present. If the PAN ID compression field is set to one, then the *device PAN ID* shall be set to the Destination PAN Identifier field. The *device addressing mode* shall be set according to the Source Addressing Mode field, as defined in Table 147. The *device address* shall be set to the Source Address, if present.

f) Obtain KeyDescriptor

The procedure shall obtain the *KeyDescriptor* using the KeyDescriptor lookup procedure as described in 9.2.2 with using the *key identifier mode* as KeyIdMode, *key index* as KeyIndex, *key source* as KeySource, *device addressing mode*, *device PAN ID*, and *device address*. If that procedure fails the procedure shall return with a *status* of UNAVAILABLE_KEY.

g) Get DeviceDescriptor

The procedure shall obtain the *DeviceDescriptor* using the DeviceDescriptor lookup procedure described in 9.2.4. If that procedure fails, then the procedure shall return with a *status* of UNAVAILABLE_DEVICE.

h) Process Frame Counter

If not using TSCH mode, the procedure shall set *frame counter* to the Frame Counter field of the *frame to be unsecured*. If *frame counter* has the value 0xffffffff, or if the *frame counter* is less than the FrameCounter element of the *DeviceDescriptor*, the procedure shall return with a *status* of COUNTER_ERROR.

i) Unsecure frame

For frames specified in the Table 146 the *Private Payload field* and *Open Payload field* shall be set as indicate there. For frames not specified in Table 146 the *Private Payload field* shall be set to contain all MAC payload fields, and the *Open Payload field* shall be empty. The procedure shall then use the *Private Payload field*, the *Open Payload field*, the ExtAddress element of the *DeviceDescriptor*, *frame counter*, the *security level*, and the Key element of the *KeyDescriptor* to produce the *unsecured frame*, according to the CCM* inverse transformation process described in the security operations, as described in 9.3.5. If the CCM* inverse transformation process fails, the procedure shall return with a *status* of SECURITY_ERROR.

j) Get SecurityLevelDescriptor

The procedure shall obtain the *SecurityLevelDescriptor* by passing the frame type and, if the frame is a MAC command, the Command Identifier, to the SecurityLevelDescriptor lookup procedure described in 9.2.5. If that procedure fails, the procedure shall return with a *status* of UNAVAILABLE_SECURITY_LEVEL.

k) Check Security Policy

The procedure shall determine whether the *frame to be unsecured* conforms to the security level policy by passing the *SecurityLevelDescriptor* and the *security level* to the incoming security level checking procedure, as described in 9.2.6. If that procedure returns with a failed status, the procedure shall return with a *status* of IMPROPER_SECURITY_LEVEL.

l) Key Usage Policy

The procedure shall determine whether the *frame to be unsecured* conforms to the key usage policy by passing the *KeyDescriptor*, the frame type, and, if the frame is a MAC command, the Command Identifier field, to the incoming key usage policy checking procedure, as described in 9.2.7. If that procedure fails, the procedure shall return with a *status* of IMPROPER_KEY_TYPE.

m) Store frame counter

If not using TSCH mode, the procedure shall increment *frame counter* by one and set the FrameCounter element of the *DeviceDescriptor* to the resulting value.

n) Return SUCCESS

The procedure shall return with a *status* of SUCCESS.

9.2.3b Incoming frame security procedure for security level zero frames

This procedure is used to process the frames which has security level of zero. The input to this procedure is the *frame to be unsecured*. The outputs from this procedure are the *status* of the procedure and, if this *status* is SUCCESS, the *unsecured frame*.

The incoming frame security procedure for security level zero frames involves following steps:

a) Check for macSecurityEnabled

if the *macSecurityEnabled* attribute is set to FALSE, the procedure shall set the *unsecured frame* to be the *frame to be unsecured* and return with a *status* of SUCCESS.

b) Get Source Address

See identically named step of the section 9.2.3.

c) Get DeviceDescriptor

The procedure shall obtain the *DeviceDescriptor* using the DeviceDescriptor lookup procedure described in 9.2.4 using *security level* set to zero. If that procedure fails, then the procedure shall return with a *status* of UNAVAILABLE_DEVICE.

d) Get SecurityLevelDescriptor

See identically named step of section 9.2.3.

e) Check Security Policy

The procedure shall determine whether the *frame to be unsecured* conforms to the security level policy by passing the *SecurityLevelDescriptor* and the *security level* of zero to the incoming security level checking procedure, as described in 9.2.6. If that procedure returns with a failed status, the procedure shall return with a *status* of IMPROPER_SECURITY_LEVEL.

f) Check for Exempt

If the incoming security level checking procedure of step e) had as output the ‘conditionally passed’ status and the Exempt element of the *DeviceDescriptor* is set to FALSE, the procedure shall return with *status* of IMPROPER_SECURITY_LEVEL.

g) Return SUCCESS

The procedure shall set the *unsecured frame* to be the *frame to be unsecured* and return with a *status* of SUCCESS.

CID1151

Tero Kivinen	INSIDE Secure	324	9.4.1.3	34	Remove extra “either”.	Replace “.. is either an incrementing shared global frame counter such as ASN.” with “... is an incrementing shared global frame counter such as ASN.”.

Accept, as described in Proposed Change column.

CID1105

Tero Kivinen	INSIDE Secure	325	9.5	39	R29 was not done properly. The table 153 still refers to table 157 (DeviceDescriptor), when it should refer to 158 (DeviceDescriptorSecLevelZero). (And the link to table 157 is broken :-). Also the table 154 DeviceDescriptorList should refer to the table 157 (DeviceDescriptor).	In table 153, change macDeviceTable to macDeviceExemptTable, and change reference to table 158 instead of 157. Change description “DeviceDescriptorSecLevelZero for each device which is allowed to be exempted for the security.” Change Type of DeviceDescriptorList to ”Set of DeviceDescriptors, as defined in Table 157.”

Accept, as described in Proposed Change column.

CID1104

Tero Kivinen	INSIDE Secure	325	9.5	46	R28 was not done. Move the macFrameCounter from the table 153 to KeyDescriptor table, i.e. table 154	Move the macFrameCounter from the table 153 to KeyDescriptor table, i.e. table 154

Accept, as described in Proposed Change column.

CID1154

Tero Kivinen	INSIDE Secure	522	23.3.2	14	The security of the fragmentation is still completely broken.	It needs to be fixed. It needs a bit different nonce generation format.

Add new subsection 9.3.2.3:

9.3.2.3 CCM* nonce for Fragments

The CCM* nonce for the fragments shall be formatted as shown in Figure xxx. The Source Address and security level as set as in defined in 9.3.2.1, and the Fragment Frame Counter is set to match the *phyFragmentFrameCounter*, and the Fragment number is set to match the fragment number of the fragment. Fragment indicator shall be set to 1.

Figure xxx – CCM* nonce for fragments

Octets: 8	Bits: 0-25	26-31	32-33	34	35-39
Source Address	Fragment frame counter	Fragment number	Security level	Fragment indicator	Reserved

We might want to split the security level field in the 9.3.2.1 to have 2 bits of security level and 6 bits of reserved just to be clear how the 2-bit security level field is formatted inside the 1-octet space reserved for it.

I am not sure whether it is good idea to refer to the phyFragmentFrameCounter here, or whether we should just use term Fragment frame counter, and modify the text in section 23.3.1 to match. The section 23.3 might need even more changes, it still needs to be checked properly.