**IEEE P802.15**

**Wireless Personal Area Networks**

|  |  |  |
| --- | --- | --- |
| Project | IEEE P802.15 Working Group for Wireless Personal Area Networks (WPANs) | |
| Title | **Suggested Resolution Security Comments on 802.15.4i, Draft D04** | |
| Date | February 16, 2011 | |
| Source | René Struik  Struik Security Consultancy  723 Carlaw Avenue  Toronto ON  Canada M4K 3K8 | E-mail: rstruik.ext@gmail.com  Phone US: +1 (415) 690-7363  Phone Toronto: +1 (416) 466-8084  Cell: +1 (647) 867-5658 |
| Re: | IEEE 802.15.4 TGi/Draft D04 | |
| Abstract | This document provides suggested resolutions of security-related comments that were received with sponsor ballot on 802.15.4i/D04, as well as a number of suggested editorial changes to improve the security-relevant portions of 802.15.4i. | |
| Purpose | Assist with fixing/improving the security-relevant portion of the draft 802.15.4i standard. | |
| 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. | |

IMPORTANT NOTE:

The suggested resolutions in this submission are relative to IEEE 802.15.4i/D04. This document will be updated, so as to provide detailed security resolutions relative to 802.15.4i/D05.**Comments on IEEE 802.15.4i, Draft D4**

1. (TR) (CID #71/D04) Clause 7.2.1, p. 141, Step f)-h): The current outgoing frame security procedure does not check whether so-called “frame counter role-over” may have occurred. If frame counter roll-over indeed occurred, then reuse of the counter would completely break cryptographic security of crypto construct, due to nonce reuse (which is obviously not intended). **Suggested remedy:** implement this check via a corresponding Blacklisted element. For details, please see 09/829r0, Clause 7.5.8.2.1, Step h) and Clause 7.6.1 hereof (adding corresponding Security-Related MAC PIB attribute Blacklisted element in Table 91). Note RS: unfortunately, this results in some reorganization of MAC PIB attributes and procedures. BTW – Clause 7.5.8.2.x in 09/829r0 corresponds to Clause 7.2.x in 802.15.4i/D4.

Changes:

* Clause 7.2.1: changed Step g), which now yields KeyDescriptor and allows indexing of Blacklisted element hereof later on.
* Clause 7.2.1: inserted Step g2) as Step h), which allows inspecting of Blacklisted element.
* Clause 7.2.1: inserted Step j2) as Step l), which sets the Blacklisted element if the frame counter has rolled over (this is not done for all other keys, since that was considered “nannying” by some TG4b people at the time, although it probably should, since each device has only one frame counter, irrespective of the number of keys stored). Note, however, that outgoing frame security with that frame counter will fail (Step f) from now on, unless the frame counter is reset by some TG4i-external process. Thus, all other keys could in theory be blacklisted via this external process then. Note RS: personally, I would rather include this safety precaution in the specification (TO BE DISCUSSED).
* Clause 7.2.2: The outgoing frame key retrieval procedure now yields a KeyDescriptor, rather than just a key, thus allowing subsequent inspection of, e.g., the Blacklisted element.
* Clause 7.2.2: This procedure now looks up the KeyDescriptor in two stages, via look-up of KeySourceDescriptor (a new procedure, inserted right after Clause 7.2.7 [DeviceDescriptor lookup procedure]), and by searching on the proper key index. This brings this more in line with the incoming frame security material retrieval procedure of Clause 7.2.4 and with the Key Identifier field.
* Clause 7.5, p. 154: introduced key source table.
* Clause 7.5.1, Table 58: added macKeySourceTable.
* Clause 7.5.1, right after Table 64: introduced Table 65 – elements of KeySourceDescriptor.
* Clause 7.5.8, p. 159: inserted Clause 7.5.8a, with description key source table.
* Clause 7.2.5: The KeyDescriptor lookup procedure now yields, indeed, the KeyDescriptor, if there.

1. (TR) (CID #72/D04) Clause 7.2.11, p. 147-148: The current security level checking procedure accepts incoming frames with a security level that is greater than or equal to a particular minimum security level (as defined in Table 92 – SecurityLevelDescriptor, p. 209). In particular, if this minimum security level is set to zero, this allows receipt of frames that are protected with confidentiality only and without authenticity (security level 0x04). Unfortunately, this may have as side effect that one can manipulate the frame counter entry of the sending device as stored on the recipient device and set this to any value (including 0xffffffff, thereby disabling further communication from that device). This can be prevented by always only allowing secured traffic, but this would hamper flexibility (since now joining devices always have to use Exempt flags and, e.g., unsecured association commands open this up to vulnerabilities). This clearly was not intended. To further illustrate the usefulness of implementing a set of acceptable security levels, rather than the notion of minimum security level (such as 802.15.4-2006 currently has), one may wish to allow data authenticity, but no encryption, as, e.g., may be the case due to regulatory constraints (no encryption allowed policies [e.g., with telco applications running over 802.15.4]), which currently cannot be expressed via the specification. As another example, one may wish to make the protection requirements dependent on the purported originator of the frame (currently, one cannot discriminate w.r.t. originator of the frame). The latter may help with the joining process, where one only allows unsecured communication from the joining node for a limited time window. NOTE – This construct has been partially implemented with 802.15.4i/D4, e.g., Clause 7.2.11 refers to SecurityModeDescriptor, but the latter has not been properly defined in Clause 7.5 (e.g., Table 62 still uses the supposedly now defunct SecurityMinimum element. **Suggested remedy:** replace the minimum security level by a set of security levels allowed. Note RS: for details, please see 09/829r0, Clause 7.5.8.2.3, in particular Step g), 7.5.8.2.11, and Table 92, Table 95 – SecurityModeDescriptor (which all implement a set, a membership test on a set). Please note that the current minimum security level test is a special instantiation of this more general set membership test, thus assuring compatibility. BTW – Clause 7.5.8.2.x in 09/829r0 corresponds to Clause 7.2.x in 802.15.4i/D4. (NOTE RS: As an aside, the risk of unlimited increments of frame counters is aleviated once the timeliness property is implemented with the specification (i.e., rejection of stale frames). This is however not part of TG4i’s scope, but will be implemented with TG4e.)

Changes:

* Clause 7.5.1, Table 58: With description of *macSecurityLevelTable*, replace “minimum security level” by “set of security levels”.
* Clause 7.5.4, Table 62: Change this table to reflect a set of security levels, rather than a minimum. The corresponding SecurityModeDecriptors are inserted at the end of security-related tables (i.e., after Table 64 in draft). (With the current draft D4, the SecurityModeDescriptor is referred on in Clause 7.2.11, but missing.) Note RS: for now, I left the PIB entry *SecurityMinimum* in (even though this does not seem to make sense to me), since that was part of stated resolution of TG4i BRC conf call as of Fri February 11, 2011.
* Clause 7.5.4, p. 158: Replaced title “minimum security level table” by “security level table”, made corresponding change at beginning of Clause 7.5, and removed “minimum” from subsequent text.

1. (TR) (CID #73/D04) Clause 7.2.3, p. 143, Step f): The current incoming frame security procedure does not properly treat devices with so-called diplomatic immunity status (Exempt status), since one never gets into checking this status if the security level is set to zero (cf. Step f), resp. i)). This prevents the main use case for this Exempt status flag, viz. temporarily allowing unsecured frames for devices in the process of joining a network (and, thereby, prior to obtaining keying material). NOTE – This has been partially implemented with 802.15.4i/D4, Clause 7.2.3, but the corresponding changes to the subroutines (e.g., Clause 7.2.4, 7.2.5, 7.2.7, etc.) have not been implemented. As an example, Step g) intends to obtain the DeviceDescriptor via 7.2.5, but 7.2.5 currently yields the KeyDescriptor. This all obviously need to be fixted. **Suggested remedy:** Implement this properly, as specified in 08/829r0, Clause 7.5.8.2.3, in particular Step h) to Step k). Note RS: unfortunately, this seemingly results in massive changes, due to need to untie some of the procedures. Easiest way would be to replace the entire clause by the one stipulated with 09/829r0.

Changes:

* First some background: With 802.15.4-2006, an unsecured frame would be accepted if the security policy check would result in pass (security level allowed) or conditionally pass (security level normally not okay, but DeviceOverrideSecurityMinimum flag set). This is not granular enough, since it would allow any device to either by-pass normal security and not just the happy few that have explicitly been granted this diplomatic immunity status (what the Exempt flag in the DeviceDescriptor was supposed to achieve). As an example, if a device comes out of the box and tries and join the network without any keys, then this Exempt flag setting for association commands and data frames may be useful, so that higher layer protocols can set-up a key first (after which the diplomatic immunity for the joining device is revoked and it has to stick to the rules from now on). With TG4i/Draft D4, this is not the case any more (thanks to Step f). Nevertheless, one still has to dig up the Exempt flag of the DeviceDescriptor, without tieing this to keys, etc. (as the 802.15.4-2006 spec did), so as to check whether the specific originating device in question had Exempt status or not (this is what Step h does after the fixes below). If so, the unsecured frame would be accepted; if not, it would be rejected. The remainder of the incoming frame security processing (Step i) onwards) would then deal with the ordinary cases (no diplomatic immunity, real security applied, etc.).This requires untangling the procedure in 802.15.4-2006, 7.5.8.2.3, Step g, and getting a DeviceDescriptor independently of KeyDescriptor and KeyDeviceDescriptor. Thus, the comment and reference to seemingly massive changes (due to the untieing). In the end, this is not as bad as it seems, but may show as a big change bar nevertheless.
* Clause 7.2.3, p. 143, Step g, l. 25: the procedure in 7.2.5 is actually not the right procedure, since it has the wrong name and the output should be the DeviceDescriptor. Introduced the currently missing incoming frame device retrieval procedure as new procedure (cf. also 09/829r0, Clause 7.5.8.2.5).
* Clause 7.5, p. 156, Table 59: Removed the topmost two entries, since not required any more due to unentanglement of KeyDescriptor, DeviceDescriptor, and KeyDeviceDescriptor procedures. Similarly, one can remove Table 64 (roughly replaced by KeySourceDecriptor – cf. 09/829r0, Table 94).
* Clause 7.5, Table 61, p. 157: Removed the topmost two entries DeviceDescriptorHandle and UniqueDevice, since never invoked after edits. Inserted DeviceAddress instead (cf. 09/829r0, Table 91, p. 214).
* Clause 7.2.7, p. 146: This procedure should have as input the device lookup data and device lookup size, as determined in the incoming frame device retrieval procedure, and yield as output the DeviceDescriptor, if there. Made fixes as also indicated in 09/829r0, Clause 7.5.8.2.8.
* Clause 7.2.3, p. 146, Step k): Again, a mismatch here, since 7.2.4 is not the frame key retrieval procedure and output is KeyDescriptor and lots of other stuff. Corrected version would be the one of 09/829r0, Clause 7.5.8.2.4 (which was half-copied only, hence all the errors here). Note that the resulting Clause 7.2.2 and 7.2.4 are almost identical and use the same subroutines (thus, providing additional rationale for editing Clause 7.2.2).
* Clause 7.2.3, p. 146, Step l): Again, a mismatch here, since 7.2.7 is the not the KeyDeviceDescriptor lookup procedure and output is not a KeyDeviceDescriptor. Corrected version would be the one of 09/829r0, Clause 7.5.8.2.7. (which seemed to have been copied only half-way, hence all the errors here).
* Clause 7.2.6, p. 146: This procedure should be removed, since absorbed by the KeyDeviceDescriptor procedure (Clause 7.5.8.2.7 of 09/829r0) and the test on the Blacklisted element performed in Step l) of 7.2.3.
* Clause 7.2.3, p. 146: I moved the frame counter steps (Step i) and Step j)) to right after Step m) {the key usage policy check}. Motivation: with TG4e, one transforms the frame counter and requires the key (now in Step k), so to keep this all in a logical flow, it is better to change the order somewhat. This has no impact on behavior, but avoids cumbersome spaghetti code.

General note RS: please check reference to Annexes (right-concatenation, etc.), since now link is broken.

Other detailed edits below (outside sponsor ballot realm):

1. (T) Clause 7.3.2, p. 148, l. 39: Replace “in B2.2” by “in B3.2”. **Suggested remedy**: implement accordingly.
2. (T) Clause 7.3.2, p. 148, l. 52: Replace “see 7.2.2” by “see 7.4”. **Suggested remedy:** implement accordingly.
3. (E) Clause 7.3.3, p. 149, l. 7: Replace “CCM\* decryption and authentication checking process” by “CCM\* decryption and authentication checking transformation”. **Suggested remedy:** Implement accordingly.
4. (E) Clause 7.3.3, p. 149, l. 11: Replace “bit ordering” by “integer and octet ordering”. **Suggested remedy:** implement accordingly.
5. (E) Clause 7.3.3, p. 149, l. 7-15: This text can be deleted entirely, since a repeat of Annex B3.2. **Suggested remedy:** implement accordingly.
6. (TR) Clause 7.3.4.2, Table 53, pp. 149-150: The Auxiliary Security Header field is technically part of the MHR (cf., Clause 5.2.1, Fig. 30, p. 53) and it is obviously not the intention to copy that substring twice in the ‘a data’. Intention is as follows: Let STR be the right-concatenation of the MHR and MAC payload fields of the frame to be transmitted. The ‘m data’ now is the Payload field of the MAC payload in case the frame requires encryption and the empty string otherwise. The ‘a data’ is the unique string so that the right-concatenation of ‘a’ and ‘m’ is the string STR. **Suggested remedy:** implement accordingly (or use alternative equivalent formulation). Rewrite this as follows: “The m data field shall be set to the (unsecured) Payload Field of the MAC Payload if frame security includes providing confidentiality and shall be set to the empty string otherwise. The a data field shall be set to the unique string so that the right-concatenation of the a data field and the m data field is equal to the MAC frame, with the FCS field purged.
7. (TR) Clause 7.3.4.2, l. 20-21, p. 150: This incorrectly delves into internal details of the cryptographic mode of operation CCM\*. The external interface only involves strings a, m, and c, \*not\* an authentication tag or encryption operation (both ‘under the hood’). More importantly, though, the statement is incorrect for frames that are authenticated, but not encrypted: for those frames, the output ‘c’ (which is the authentication tag) of the CCM\* forward operation is “inserted” at the right of the unsecured payload field, i.e., the secured payload field is the right-concatenation of the unsecured payload field and the string ‘c’ . **Suggested remedy:** Rewrite this sentence as follows: “The Payload field of the MAC Payload shall be set to the string ‘c’ if frame security includes providing confidentiality and shall be set to the right-concatenation of the (unsecured) Payload field of the MAC Payload and the string ‘c’ otherwise.” Note RS: a similar change needs to be made with the outgoing frame security procedure (Clause 7.2.1, Step i), since also not entirely correct.
8. (TR) Clause 7.2.1, p. 141, Step i), l. 26: Technically, encryption is applied to the string ‘m’ and the authentication tag alike. In particular, even if only authentication is applied, then the encryption transformation of Annex B4.1.3 is still applied, but only over the authentication tag. **Suggested remedy:** Rewrite this so as to be consistent with Clause 7.3 (Security operations). In fact, why not simply remove l. 23-34 (Sub-steps 1-3) entirely, since already spelled out in Clause 7.3.4?
9. (T) Clause 7.3.4.2, Table 54, p. 150: The terms “secured” (and “unsecured”) are too loosely defined here, since “secured” means “encrypted” here, but generally may include anything resulting from applying security services (including authentication [in which case the MIC field is inadvertently referred to twice in the table]). **Suggested remedy:** Replace “secured” by “encrypted”.
10. (T) Clause 7.3.4.2, Table 54: p. 150: The authentication tag is actually an \*encrypted\* authentication tag (if one looks under the hood of the CCM\* forward operation), no matter whether confidentiality over the outgoing frame is provided or not. **Suggested remedy:** Avoid this altogether by just referring to parameters that show up in the external interface of the CCM\* mode of operation, rather than diving “under the hood”.
11. (T) Clause 7.3.5.2, Table 55, p. 151:The terms “secured” (and “unsecured”) are too loosely defined here, since “secured” means “encrypted” here, but generally may include anything resulting from applying security services (including authentication [in which case the MIC field is inadvertently referred to twice in the table]). **Suggested remedy:** Replace “secured” by “encrypted”.
12. (T) Clause 7.3.5.2, Table 55: p. 151: The authentication tag is actually an \*encrypted\* authentication tag (if one looks under the hood of the CCM\* forward operation), no matter whether confidentiality over the outgoing frame is provided or not. Moreover, when encryption was not applied to the frame, the “secured” payload field (minus the authentication tag) is actually just the same as the unsecured payload field. **Suggested remedy:** Avoid this altogether by just referring to parameters that show up in the external interface of the CCM\* mode of operation, rather than diving “under the hood”.
13. (TR) Clause 7.3.5.3, p. 151, l. 38-39: This incorrectly delves into internal details of the cryptographic mode of operation CCM\*. The external interface only involves strings a, m, and c, \*not\* an authentication tag or encryption operation (both ‘under the hood’). More importantly, though, the statement is incorrect for incoming frames that were authenticated, but not encrypted: for those frames, the input ‘c’ (which is the authentication tag) of the CCM\* inverse operation is “purged” from the unsecured payload field, i.e., the unsecured payload field results from removing the rightmost substring ‘c’ from the received payload field. **Suggested remedy:** Rewrite this sentence as follows: “The unsecured payload field of the MAC Payload shall be set to the string ‘m’ if frame security includes providing confidentiality and shall be set to the (secured) Payload field of the MAC Payload, with the rightmost substring ‘c’ purged, otherwise.” Note RS: a similar change needs to be made with the incoming frame security procedure (Clause 7.2.3, Step n), since also not entirely correct.
14. (TR) Clause 7.2.3, p. 141, Step n), l. 7: Technically, decryption is applied to the string ‘c’, which includes the authentication tag (irrespective of whether encryption was supposedly applied). In particular, even if only authentication was applied, then the encryption transformation of Annex B4.1.3 is still applied, but only over the authentication tag. **Suggested remedy:** Rewrite this so as to be consistent with Clause 7.3 (Security operations). In fact, why not simply remove l. 3-15 (Sub-steps 1-3) entirely, since already spelled out in Clause 7.3.5?
15. (TR) Clause 7.2, p. 140, l. 21: There intended behavior when the *macSecurityEnabled* attribute is set to FALSE is not specified. **Suggested remedy:** Replace “with security when” by “with security only if”.
16. (TR) Clause 7.2.1, p. 141, l. 2: For consistency of style, one should replace “return with the secured frame and a status of SUCCESS” by “return with a status of SUCCESS”. **Suggested remedy:** Implement accordingly.
17. (TR) Clause 7.2.8, p. 146: This procedure has the same title as that of Clause 7.2.11. **Suggested remedy:** correct accordingly.
18. (E) Clause 7.2.3, p. 142, l. 38: The output of the incoming frame security procedure is not in line with what is done elsewhere. As an example, with 7.2.2 and elsewhere, one has something along the following lines: “the outputs from this procedure are a passed or failed status and, if passed, …”. **Suggested remedy:** Use similar language for similar cases (parallelism of style), as suggested.
19. (E) Clause 7.2.3, p. 142, l. 45: should one or shouldn’t one add “as applicable” (as does Clause 7.2.1, p. 140, l. 33)? Not sure what “as applicable” means here, since intent clearly is that all steps are executed and not just a subset to one’s liking and that only exits the routine prematurely if an error condition has been met. (Also elsewhere). **Suggested remedy:** Again, consistency of style is key here, as suggested.
20. (TR) Clause 7.2.3, p. 142, l. 42-43: This statement seems more a configuration remark and seems to be redundant. If not redundant, the statement does not seem to be complete, since it does not make similar assumptions on frame counters, short addresses, security level policy settings, etc. – which should also all be configured. **Suggested remedy:** Remove this sentence, make this a Note, or otherwise, and address incompleteness remark illustrated in comment.
21. (TR) Clause 7.2.3, p. 143, Step g): The DeviceDescriptor cannot be obtained via a procedure call to Clause 7.2.5, since the latter yields a KeyDescriptor and not a DeviceDescriptor. **Suggested remedy:** Fix in line with 09/829r0.
22. (E) Clause 7.2.3, p. 143, Step h, l. 29-30: To be consistent with style elsewhere, shorten this sentence by stipulating “return with a status of SUCCESS, if <Condition A>, and with a status of <Stat-Code-B> otherwise. **Suggested remedy:** Implement accordingly.