IEEE P802.15

Wireless Personal Area Networks

|  |  |  |
| --- | --- | --- |
| Project | IEEE P802.15 Working Group for Wireless Personal Area Networks (WPANs) | |
| Title | Privacy primitives | |
| Date Submitted | 12th March 2024 | |
| Source | Tero Kivinen | E-mail: kivinen@iki.fi |
| Re: | TG4ac draft | |
| Abstract | Primitives needed to provide privacy on 802.15.4 | |
| Purpose | Create TG4ac draft | |
| 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. | |

1. Definitions, acronyms, and abbreviations
   1. Definitions

**device identifier:** Random 64-bit identifier for the device.

**extended privacy address:** Random 64-bit address in form of AAI-64.

**network identifier:** Random 64-bit identifier to identify the network.

**network key:** Key shared between all devices in the network.

* 1. Acronyms and abbreviations

DI device identifier

LMO left most octet

SANGP short address nonce generation prefix

RMO right most octet

1. Optional features

10.9a Privacy enhancements

1. General

This optional feature tries to provide protection against tracking of the devices, i.e., tracking the location of the device as it moves around and using that to find out the movements of the person carrying that device. Because different devices in the same network might have different privacy requirements, and it is up to the device needing strict privacy protection to drive the process to provide or require it. In common use case this is for example the mobile phone carried by the person, thus the device needs protection against tracking, and requires use of addresses that cannot be tracked, and it needs to change addresses frequently enough so that tracking of the user is not feasible.

Complete protection against tracking of individual user is impossible, but the enhancements here provide methods to block mass surveillance and tracking of users. National state attackers could still for example store the radio fingerprint of the device to be tracked, and use special radios to keep track of users location, but this is not something that can be scaled to keep track of all users.

Protection against tracking individual users done by other users is still in scope of this enhancements, i.e., it tries to provide protection against attacker using generally available hardware to keep track of user.

Fixed installations like light switches and fixtures do not have privacy requirements for themselves, but they need to cope with the device that have, thus they need to implement this enhancement enough to be able to allow other devices connecting to them to have privacy addresses, and they need to be able to understand that device connecting them do not want to keep stable static address for them.

Privacy enhancements require the use of encryption and authentication as defined in section 9 to protect the message exchange between devices. IEEE Std 802.15.9 may be used to generate keys for the security layer. All MAC commands described in this section shall be sent in encrypted and authenticated frames. The IEs described in this section contain separate protection inside and may be also sent in frames which are not encrypted and/or authenticated.

1. Identifiers
   1. Generic

All IDs and addresses used in the privacy enhancement feature use address formats defined in the IEEE Std 802c-2017 section 8.4.4.3, but use the transmission order defined in the section 4.5.1. This enhancement adds two more bits (S, and T) to the format defined in the IEEE 802c.

Figure 1 shows the location of the S, T, X, Y, Z and M bits in the transmission of the address C2-01-23-45-67-89-AB-CD.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| RMO→LMO | Octet 8 RMO | Octet 7 | Octet 6 | Octet 5 | Octet 4 | Octet 3 | Octet 2 | Octet 1 LMO |
|  | CD | AB | 89 | 67 | 45 | 23 | 01 | C0 |
| LSB→MSB | 10110011 | 11010101 | 10010001 | 11100110 | 10100010 | 11000100 | 10000000 | 11000000 MXYZST |
| Transmission order | | | | | | | | |

Figure 1—Location of the M/X/Y/Z/S/T fields

M bit has its normal meaning, i.e., it defines whether the address is unicast or multicast address. The X, Y, and Z bits shall be set to one, zero, zero, respectively to indicate an Administratively Assigned Identifier (AAI-64) addresses for all extended addresses used by this enhancement. The next two bits S and T identify the type of the address to be used. For the extended address used in the MHR over the air the S and T bit shall be set to zero which makes sure it does not collide with multicast addresses defined by IETF RFC 2464. For other uses see Table 1.

Table 1—Usage of S and T bits

|  |  |  |  |
| --- | --- | --- | --- |
| **S** | **T** | **Usage** | **Description** |
| 0 | 0 | Extended privacy address | Extended address used in the frames. This is only value used over the air in clear. |
| 0 | 1 | Device identifier (DI) | DI is used to identify the device and is always sent in frames that are encrypted, thus it will not appear over the air in clear. |
| 1 | 0 | Network ID | Network ID is only used internally to identify the network, and it can be used to generate the network key, but is not otherwise sent over the air. |
| 1 | 1 | Short address nonce generation prefix (SANGP) | Prefix to be used when generating nonce for short addresses. This value may overlap with the IETF RFC 2464 which uses 33-33 for the first two octets, so it can not be used in MHR. |

* 1. Network ID

Network ID is a random ID used to identify network. Privacy enhancements feature described in this section never sends this inside any frames, but next higher layer may need to send this over air while provisioning devices or when device first time joins the network. In such cases when network ID is included in any frame, the frame shall be encrypted. Network ID does not change during the lifetime of the network. Different networks should use different network IDs, and there is one “owner” of the network ID, i.e., the one who announces the network using the network ID.

When the device creates a network, it will generate random network ID and other devices joining the network will need to get this out of band or using next higher layer protocol that is outside the scope of this standard.

Network ID is an AAI-64 address as defined in section 10.9a.2.1.

* 1. Network key

Network key is the 128-bit key used to for network announcements. This can either be negotiated using IEEE Std 802.15.9 or distributed out of band, or it can also be 64-bit network ID (as a string) concatenated with 64-bits of zeros to make it 128-bit, in which case anybody who knows the network ID knows the network key, and the security is reduced to 58-bits.

NOTE—The network ID when generating network key uses the representation format described in section 9.3.1.

The format of generating network key from network ID is show in Figure 2.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Network ID Octets: 8** | | | | | | | | **Padding Octets: 8** |
| LMO Octet 1  MSB→LSB xxTSZYXM | 2 | 3 | 4 | 5 | 6 | 7 | RMO Octet 8 | 0x00 00 00 00 00 00 00 00 |

Figure 2—Generation of the network key from network ID

This key is used to authenticate the network announcements. This does not provide source authentication, thus anybody who knows the network key can claim to be the owner of the network.

* 1. Device identifier (DI)

Device identifier (DI) is a random ID used to identify device. When this is sent in any frames the frames shall be encrypted. This does not change during the lifetime of the network for the device. Device may use different DIs in different networks. This is never used in source or destination address fields of the MAC header of the frame, and it is never used when generating security nonce for the frame.

DI is an AAI-64 address as defined in section 10.9a.2.1.

* 1. Extended privacy address

To provide privacy of the device the device shall not use its stable extended address, but instead device will generate random extended privacy addresses as needed. The frequency of generation of addresses and how many addresses are used is left to the next higher layer, as different uses cases have different requirements.

For example the light switch which is in static location does not really benefit from generating privacy addresses. On the other hand light fixtures might need to use different addresses, not to protect their own privacy but to protect the privacy of the device connecting to them, i.e., when mobile phone connects to light fixture, it wants to use its own randomly generated extended privacy address as a source address, and it wants the light fixture to also have extended privacy address that is not static all the time.

The extended privacy addresses are generated by the device and device can have multiple extended privacy addresses for the same network. A device should use a different set of extended privacy addresses for each network it is connected to. It may use a different set of extended privacy addresses when communicating with different devices in the same network.

When it stops advertising some extended privacy address to a peer, the peer is allowed to remove that extended address from its security tables, thus device removing the address is not allowed to use that address anymore with that peer.

NOTE—As the peer has removed extended privacy address from its security tables, it has also removed the *secDeviceMinFrameCounter* thus if device would start using the same address again it would allow replay attacks.

The probability of generating same random extended privacy address twice is so small, that it can be ignored when generating addresses, so even when devices not allowed to reuse same extended privacy addresses they have once stopped using there is no requirement for remembering all extended privacy addresses device has used before.

Extended privacy address is an AAI-64 address as defined in section 10.9a.2.1.

Each extended privacy address shall have separate entries in the *secDeviceList* and in the *secKeyDeviceFrameCounterList* element of the *secKeyDescriptor* structures. This means each of the extended privacy addresses shall have separate frame counters.

* 1. Short addresses

Short address are already locally allocated and are not stable, so using them will provide limited privacy. The privacy enhancements option adds option to have multiple short addresses to identify same device, and to be able to assign set of addresses to devices and reassign device a new set of addresses. The number of addresses assigned and the frequency of their reassignment is left to the next higher layer of the device assigning those addresses.

Each short address is associated with a PAN ID, and this section provides a way to use different PAN IDs associated with different set of addresses as needed. Next higher layer assigning addresses may use one PAN ID at time, and assign addresses to each device in the network using that PAN ID, and then change this PAN ID by assigning new set of addresses to each device from new PAN ID. Or it may assign different PAN IDs and lists of short addresses to different devices. Between pair of devices only one PAN ID is used at one time.

The next higher layer shall generate short addresses in such way that they are unique for each key used in the security layer, i.e., even if device assigns different PAN IDs and different set of short addresses to two different devices, if those devices share a key those short addresses shall be generated in such way they are all unique, even when they are in different PAN ID.

To be able to use short addresses for encrypted frames the receiving device needs to know the extended address of the sender to be used for nonce generation as described in section 9.3.2.1. This is done by using a short address nonce generation prefix (SANGP) concatenated with short address to generate unique extended address for each short address for the nonce generation purposes. Replay protection counters are associated with this generated extended address.

* 1. Short address nonce generation prefix (SANGP)

To generate unique extended address for nonce generation for each short address, each device uses 48-bit SANGP. This prefix is used to generate the extended address for nonce generation when using short addresses as shown in figure 3.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Octets: 6** | | | | | | **2** |
| **SANGP** | | | | | | **Short address** |
| LMO Octet 1  MSB→LSB xxTSZYXM | 2 | 3 | 4 | 5 | RMO Octet 6 |

Figure 3—Generation of the extended address used in the nonce when using short addresses.

This means each short address has its own unique extended address used for nonce generation, and each short address will also have separate frame counter and separate *secDeviceDescriptor* entry in *secDeviceList*. Because each short address has separate *secDeviceDescriptor* entry it means the *secDeviceMinFrameCounter* is also separate if *secFrameCounterPerKey* is set FALSE. If *secFrameCounterPerKey* is TRUE for the key used then each key will have its own frame counters and each short address shall have its own *secKeyDeviceFrameCounterDescriptor* entry in the *secKeyDeviceFrameCounterList* of the key.

Each device assigning short addresses shall also distribute SANGP associated with the list of short addresses for nonce generation purposes. SANGP shall be generated in such way that no duplicate values are generated. Generating random 42-bit value and adding fixed bit prefix to it is acceptable solution for small networks (when there is no more than few thousands devices connected to that network over the lifetime of the key used in the network). In larger networks the entity can simply use 42-bit counter and assign sequential SANGPs to devices, or use some other method that allocates SANGPs in a way no duplicates are generated.

SANGP is an 48-prefix from the AAI-64 address space as defined in section 10.9a.2.1.

* 1. Static extended address

The privacy enhancements option requires that device do not respond to their own static extended address after initial deployment. During the initial deployment the static extended address of the device can be used, but after that the device should not respond to any frames sent that address, also the device should not use its own static extended address as source address.

In some use cases there might be uses where device still wants to use the static addresses, for example the light fixture might respond to its own static extended address to allow new devices to join the network. This does not cause privacy issues as the light fixture in fixed location does not really have privacy requirements for itself, it only needs to do privacy enhancements to be able to cope with other devices which do have privacy requirements.

* 1. Address list sequence number

Address list sent by the device may be associated with the address list sequence number. This number is needed in case the devices use multiple extended privacy addresses at the same time, and it is used to provide ordering for the address lists received in the Address List command. If only one extended privacy address is used, or if short addresses are used, then the replay prevention mechanism provided by the security layer prevents replays of old Address List commands.

In case the device has multiple extended privacy addresses and those are used to send Address List commands, the attacker could take one using extended privacy address 1 and make sure it never reaches the recipient, and then when device moves to use extended privacy address 2 later, but still keeps extended privacy address 1 in use, the attacker replays the old frame stored before to the device, and as it is not replay (device never received it) the security layer does not drop it, and it is passed to the next higher layer. Next higher layer can then detect this situation by checking the address list sequence number in Address List commands. If the Address List Sequence Number field of the received Address List command is smaller (using sequence number arithmetic as defined in RFC1982) then the Address List command is old, and command shall be dropped.

* 1. Frame counters

Normally there is only one *secFrameCounter* for a device, or one *secKeyFrameCounter* per key, but to provide privacy when using multiple source addresses, each source address needs to have separate frame counters. This means that *secFrameCounter* and *secKeyFrameCounter* in *secKeyDescriptor* structure shall be replaced with separate counters for each source address to be used, regardless whether it is extended privacy address, or short address. In case of short addresses the addresses used are generated as described in section 10.9a.2.7. Those data structures are described in the section 10.9a.2.11.

Each *secDeviceFrameCounter* value is initialized to random value when entries are added to the security tables, and when the entry is no longer in use it may be removed.

* 1. Data structures
     1. Generic privacy-related MAC PIB attributes

The privacy-related PIB attributes are defined in Table 2.

Table 2—Privacy-related MAC PIB attributes

|  |  |  |  |
| --- | --- | --- | --- |
| **Element** | **Type** | **Range** | **Description** |
| *secFrameCounterList* | List of *secFrameCounterDescriptor* as defined in Table 3. | — | A list of *secFrameCounterDescriptor* structures containing frame counters for each source address. |
| *priNetworkList* | List of *priNetworkDescriptor* as defined in Table 5. | — | A list of *priNetworkDescriptor* structures containing description of known networks. |
| *priLocalAddressesList* | List of *priLocalAddressesDescriptor* as defined in Table 6. | — | A list of *priLocalAddressesDescriptor* structures containing description of local addresses. |
| *priRemoteDiList* | List of *priRemoteDiDescriptor* as defined in Table 1. | — | A list of *priRemoteDiDescriptor* structures containing description of the remote DIs. |

* + 1. secFrameCounterDescriptor

Table 3 defines the elements of *secFrameCounterDescriptor*, which contains the outgoing frame counter for each extended address of the local device.

Table 3—Elements of *secFrameCounterDescriptor*

|  |  |  |  |
| --- | --- | --- | --- |
| **Element** | **Type** | **Range** | **Description** |
| *secDeviceExtAddress* | IEEE address | An extended  IEEE address | An local extended address of the device. |
| *secDeviceFrameCounter* | Integer | 0x00000000–  0xffffffff | An outgoing frame counter associated with the extended address. |

* + 1. secKeyDescriptor

When using multiple local addresses the table 9-10 defining secKeyDescriptor elements is annotated by adding following elements as defined in table 4.

Table 4—Annotated elements to *secKeyDescriptor*

|  |  |  |  |
| --- | --- | --- | --- |
| **Element** | **Type** | **Range** | **Description** |
| *secKeyFrameCounterList* | List of *secFrameCounterDescriptor* as defined in Table 3. | — | A list of *secFrameCounterDescriptor* structures containing list of frame counters for each source address for this key. |

* + 1. priNetworkDescriptor

List of networks known by the device is stored in the table containing entries as shown in Table 5. This table is indexed by the *priNetwork*, but when trying to verify encrypted verifier of the Net Announcement or Net Request IEs all entries in this table are considered.

Table 5—Elements of the network table

|  |  |  |  |
| --- | --- | --- | --- |
| **Element** | **Type** | **Range** | **Description** |
| *priNetwork* | IEEE address | An extended IEEE address | Network ID of the network. |
| *priNetworkKey* | Set of octets | — | Key used by this network. |
| *priSequenceNumber* | Integer | 0x00000000-0xffffffff | Latest sequence number from the announcement packet received. |

* + 1. priLocalAddressesDescriptor

List of addresses used the be local device for each network and for each remote DI is stored in the table containing entries as shown in Table 6. This table is indexed by the *priNetwork* and optionally *priRemoteDevice*.

Table 6—Elements of *priLocalAddressesDescriptor*

|  |  |  |  |
| --- | --- | --- | --- |
| **Element** | **Type** | **Range** | **Description** |
| *priNetwork* | IEEE address | An extended IEEE address | Network ID of the network that local device is part of. This can also be empty in case the device uses same DI for all networks. |
| *priDeviceIdentifier* | IEEE address | An extended IEEE address | DI of the local device in this network |
| *priRemoteDevice* | IEEE address | An extended IEEE address | DI of the remote device. This can be empty in case the device uses same DI for all remote devices in network. |
| *priExtendedPrivacyAddresses* | List of extended privacy addresses | — | List of extended privacy addresses used by local device for this network and remote device. |
| *priPanId* | Device PAN ID | 0x0000-0xffff | PAN ID associated with the short addresses. If *priShortAddresses* element is empty then this is ignored. |
| *priShortAddresses* | List of short addresses | — | List of short addresses used by the device for this network and remote device. |
| *priNonceGenerationPrefix* | 48-bit prefix | — | SANGP for short addresses. If *priShortAddresses* element is empty then this is ignored. |

* + 1. priRemoteDiDescriptor

List of addresses associated with each remote DI is stored in a table containing entries as shown in Table 1. This table is indexed by the *priNetwork* and the *priRemoteDevice*, and contains entry for each peer of the device.

Table 7—Elements of *priRemoteDiDescriptor*

|  |  |  |  |
| --- | --- | --- | --- |
| **Element** | **Type** | **Range** | **Description** |
| *priNetwork* | IEEE address | An extended IEEE address | Network ID of the network that remote device is part of. This can also be empty, in which case the same DI is assumed to work on all networks. |
| *priRemoteDevice* | IEEE address | An extended IEEE address | DI of the remote device. |
| *priExtendedPrivacyAddresses* | List of extended privacy addresses | — | List of extended privacy addresses used by remote device. |
| *priPanId* | Device PAN ID | 0x0000-0xffff | PAN ID associated with the short addresses. If *priShortAddresses* element is empty then this is ignored. |
| *priShortAddresses* | List of short addresses | — | List of short addresses used by the remote device. |
| *priNonceGenerationPrefix* | 48-bit prefix | — | SANGP for short addresses. If *priShortAddresses* element is empty then this is ignored. |

If short addresses are used then table entry also contains the PAN id and list of short addresses associated with that PAN ID. The short addresses are associated with the SANGP found in the *priNonceGenerationPrefix*.

When new update for the extended privacy addresses or short addresses is received then the table for that peer (identified by the DI and the network ID) is updated. In addition that, the *secDeviceDescriptor* table described in 9.5.8 is updated so that any entry matching the extended privacy addresses or short addresses that were removed from the DI table are also removed there, and all new extended privacy addresses or short addresses are added. The *secKeyDeviceFrameCounterDescriptor* table should be searched for any matching extended privacy addresses, and those addresses should be removed.

1. Primitives
   1. Sending list of addresses
      1. General

Sending list of address used by this device is done using Address List command.

This may be sent to either unicast or multicast address. The source address of this may either be short address, or extended privacy address.

The Address List command contains list of short address or list of extended privacy addresses or both. The recipient will replace the address list it has with the list received in the frame. If frame only contains one type of address list then only that list shall be replaced, and other list shall be left intact. To clear list of addresses the number of addresses is sent as zero.

The main uses for this command are described in following sections.

* + 1. Sending address list with confirmation

When the device wants to send updated list of addresses to the another device it can send Address List command by issuing MLME-PRIV-ADDR-LIST primitive. The MLME-PRIV-ADDR-LIST.request sends the list to the other device, and returns a MLME-PRIV-ADDR-LIST.confirm immediately after the Address List command has been transmitted.

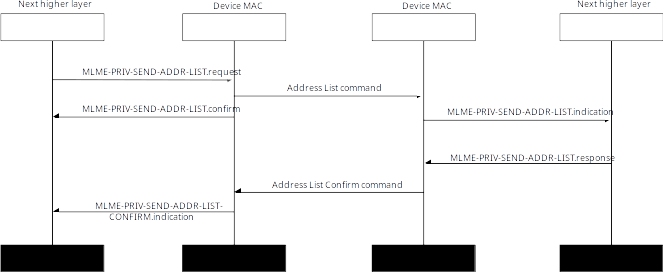
When the device receives the Address List command it will issue MLME-PRIV-ADDR-LIST.indication primitive to the next higher layer. The next higher layer may then send a confirmation message back by issuing MLME-PRIV-ADDR-LIST.response primitive.

When the MLME-PRIV-ADDR-LIST.response is issued the MAC shall send Address List Confirm command. When the device receives that frame it will issue MLME-PRIV-ADDR-LIST-CONFIRM.indication to indicate that the confirmation has been received.

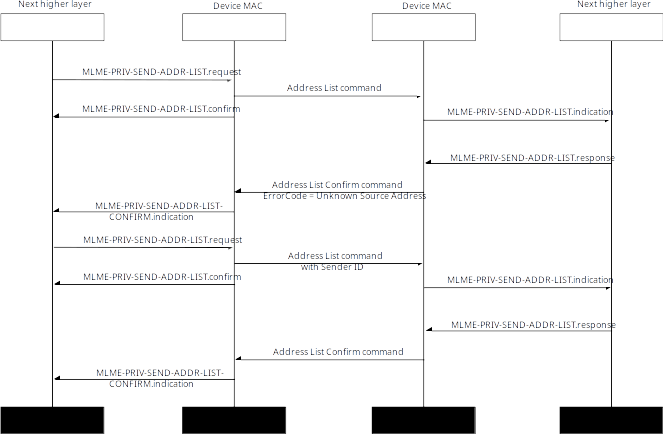
In this use case the source and destination address of the frames shall be unicast addresses of the devices. Sender ID is normally not needed as the recipient should know the sender from the source address, but it can be included in case the sender is not sure that the recipient knows its current privacy address.

Message sequence chart to send address list with confirmation is shown in Figure 4.

In case the recipient does not recognize the sender from the source address and the confirmation is required it shall send confirmation with error code Unknown Source Address, and then sending device should resend the Address List command, but include the Sender ID field to indicate sender.

Figure 4—Sending address list with confirmation

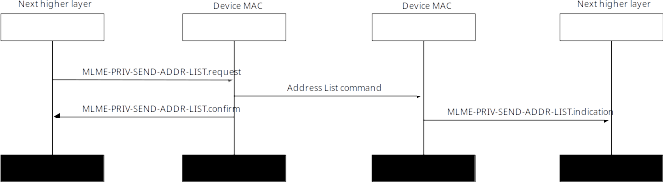
Message sequence chart of the error case where the recipient does not recognize the senders address is shown in Figure 5.

Figure 5—Sending address list and recovery after other end returns error

* + 1. Sending address list without confirmation

When device does not require confirmation from the other peer it can send the address list by using MLME-PRIV-ADDR-LIST.request and set the ConfirmationRequired parameter to FALSE.

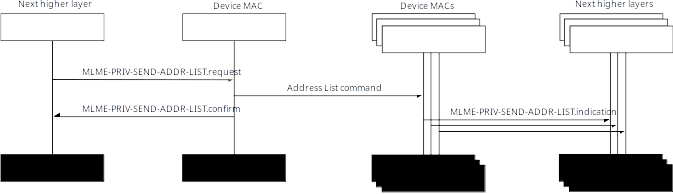
Message sequence chart to send address list without confirmation is shown in Figure 1.

Figure 6—Sending address list without confirmation

* + 1. Sending address list to multicast address

If the device wants to announce its new address to group of devices in the network it can send the Address List command to the multicast address. In that case the ConfirmationRequired shall be set to FALSE.

Message sequence chart to send address list to multicast address is shown in Figure 7.

Figure 7—Sending address list to multicast address

* 1. Confirmation of receipt of address list

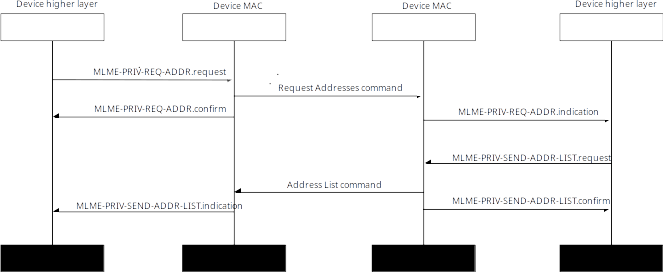
When the Address List command is received and MLME-PRIV-ADDR-LIST.indication primitive is called with ConfirmationRequested set to TRUE, the next higher layer may call MLME-PRIV-ADDR-LIST.response to request the MAC to send Address List Confirm command. This frame is always sent in unicast frame to the sender of the Address List command. This shall not be sent if the destination address of the address list was not unicast address.

This is used to confirm the reception of the Address List command.

* 1. Request to get list of addresses

If the device does not know or suspects it is out of sync with the device it has communicated before it may request a remote device to send its list of addresses by sending Request Addresses command. This command is sent to unicast or multicast address. If the device knows some address used by the remote device, it may send the command to that addresses, or if it does not know any addresses it may send command to multicast address, in which case it shall include remote peers DI in Recipient ID field. The source address is typically extended privacy address.

Typical exchange is shown in Figure 8.

Figure 8—Requesting address list from remote peer

This command may be sent after or during orphan scan, i.e., where the device things remote peer has changed address, and device do now know currently used addresses. The recipient of this will reply to that with Address List command.

1. Short address assignment
   1. Assignment of addresses to remote peer

A device in the network takes care of assigning short addresses for devices in that network. Typically this is the device who created the network, but it can also be some other entity in the network. The entity taking care of the assignments, needs to make sure that no duplicate short addresses are assigned to any devices at the same time. The entity also assigns a SANGP for each device it assigns short addresses and it shall assign SANGPs in such way that they are unique over the lifetime of the network.

Assign Addresses command is used to assign short addresses to devices. It is usually sent to the unicast address of the intended recipient, but if the transmitter thinks remote peer might be out of sync it can also send this to multicast address.

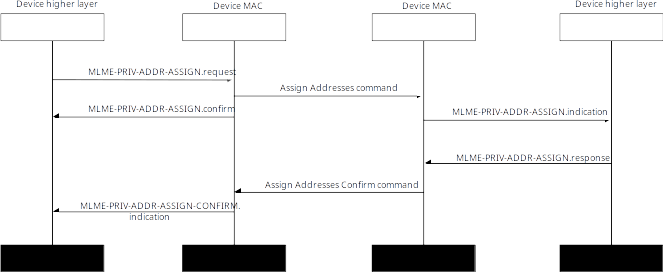
When the device wants to assign short addresses to the another device it can send Assign Addresses command by issuing MLME-PRIV-ADDR-ASSIGN primitive. The MLME-PRIV-ADDR-ASSIGN.request sends the list to the other device, and returns a MLME-PRIV-ADDR-ASSIGN.confirm immediately after the Assign Addresses command has been transmitted.

When the device receives the Assign Addresses command it will issue MLME-PRIV-ADDR-ASSIGN.indication primitive to the next higher layer. The next higher layer may then send a confirmation message back by issuing MLME-PRIV-ADDR-ASSIGN.response primitive.

When the MLME-PRIV-ADDR-ASSIGN.response is issued the MAC shall send Assign Addresses Confirm command. When the device receives that frame it will issue MLME-PRIV-ADDR-ASSIGN-CONFIRM.indication to indicate that the confirmation has been received.

In normal case the source and destination address of the frames shall be unicast addresses of the devices. Sender ID is normally not needed as the recipient should know the sender from the source address, but it can be included in case the sender is not sure that the recipient knows its current privacy address. Recipient ID is needed in case the sender sends this to multicast address, as it does not know the current address of the recipient.

Message sequence chart to send address list with confirmation is shown in Figure 9.

Figure 9—Assignment of the addresses

* 1. Confirmation of address assignment

When the Assign Addresses command is received and MLME-PRIV-ADDR-ASSIGN.indication primitive is called with ConfirmationRequested set to TRUE, the next higher layer may call MLME-PRIV-ADDR-ASSIGN.response to request the MAC to send Assign Addresses Confirm command. This frame is always sent in unicast frame to the sender of the Assign Addresses command. This shall not be sent if the destination address of the address list was not unicast address.

This is used to confirm the reception of the Assign Addresses command.

1. Updating key source
   1. General

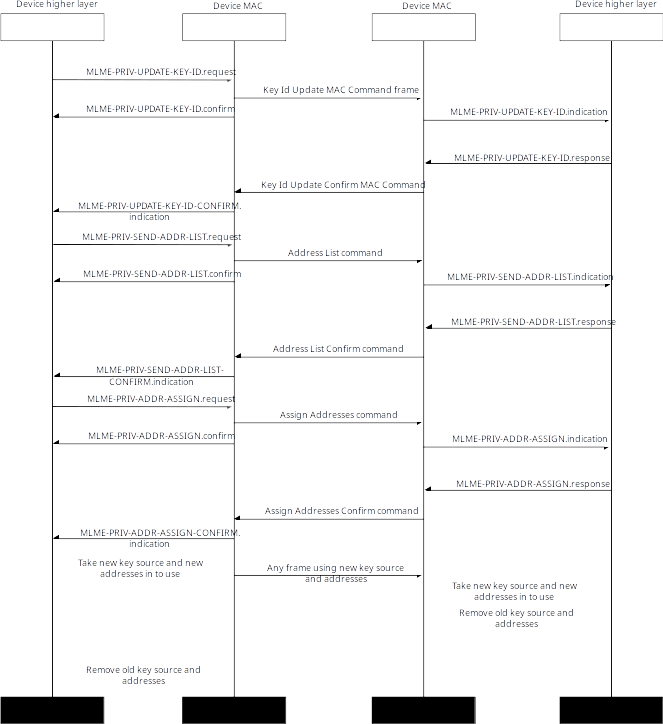
To change source and destination addresses does not help if the key source in the auxiliary security header stays same. To change key source the device can use MLME-PRIV-UPDATE-KEY-ID MLME primitives and after the key source has been changed the device shall still use the old key source to change addresses using MLME-PRIV-ADDR-LIST and MLME-PRIV-ADDR-ASSIGN MLME primitives. After both key source has been updated and new addresses advertised or assigned, then device changes to use both the new key source and new addresses at the same time. The new key source is taken in to use when the first frame is received using it, and after that the old key source can be deleted.

Changing key source only makes sense if the key identifier mode is not 0x00. When using key identifier mode 0x00 the key is identified by the source and destination addresses, thus simply changing address automatically also changes the key source. When key source is changed all the frame counters using that key source are reset to random values. After key source has been changed, device shall not use same addresses (extended privacy addresses or short addresses) it used before changing key source.

* 1. Key Identifier Update command

Key Identifier Update command may be sent as unicast or multicast message. If sent as multicast message there shall not be confirmations. Device shall first update key source and after that it shall update the addresses. After updating both key source and addresses, then transmitter shall use the new key source and addresses for its next transmission, and receiver shall take the new values in to use after receiving first frame using new key source and addresses. Device may delete the old key source and addresses from the security tables after the new values have been taken in to use, but especially if using multicast frames to send Key Identifier Update command device should keep the old values for some time in case some device in the network did not receive the update.

Message sequence chart to update key source and then sending address list and assigning new short addresses is shown in Figure 10.

Figure 10—Updating key source and addresses

* 1. Confirmation of updating key source

When the Key Identifier Update command is received and MLME-PRIV-UPDATE-KEY-ID.indication primitive is called with ConfirmationRequested set to TRUE, the next higher layer may call MLME-PRIV-UPDATE-KEY-ID.response to request the MAC to send Key Identifier Update Confirm command. This frame is always sent in unicast frame to the sender of the Key Identifier Update command. This shall not be sent if the destination address of the address list was not unicast address.

This is used to confirm the reception of the Key Identifier Update command.

1. Network discovery
   1. General

There are two generic ways of finding a known network, listening advertisements from the network or sending requests asking whether the network is nearby. The initial joining to the network is done using some out of band method, and are outside the scope of this enhancement. This section describes how device can join the network it has already been part of, i.e., it already knows the network ID and network key (the network key may be derived from the network ID).

The device may still have security context with the network, i.e., key source and key, but this is not requirement for the network discovery.

The network discovery works by generating an Network Announcement IE that contains verifier that is encrypted and authenticated using the network key. If this IE is included in the beacon or other periodic broadcast or multicast frame then devices can listen for those frames and when they receive the frame, they can try to decrypt and authenticate the encrypted verifier using the network keys they have and if they succeed, they know there is someone advertising network nearby. This does not yet mean that the advertisement is valid, as it could be a replay from the older advertisement, so next device will verify that the sequence number inside the encrypted verifier is newer than any of the sequence numbers they have seen before, and if is then they may try to join the network.

The device can also send out Net Request IE containing the encrypted verifier themselves to the broadcast or unicast address and then the network owner may contact back to the device.

* 1. Network announcement

In the Network Announcement IE case this IE is generated by owner of the network to allow devices coming to the radio range of it to join the network. This IE is contained in the beacon frame or some other periodic multicast or broadcast frame. The frame itself is usually sent without encryption, but it may contain authentication. This is because those frames are usually also used for initial joining to the network, thus devices who do not yet have security context with the network may need those frames to be able to join the network. Source address shall be an extended address (this extended address is used for the nonce generation when encrypting and authenticating the Encrypted Verifier).

The next higher layer may use MLME-PRIV-NET-VERIFIER-GENERATE MLME primitives to generate an encrypted verifier that is contained in the Network Announcement IE. That encrypted verifier is encrypted and authenticated using the network key. The encryption and authentication uses same format than what is used for when encrypting and authenticating normal frames, except the data to be encrypted and authenticated only contains Announcement Nonce and Sequence Number fields, and the nonce is generated from the extended address from the MHR, and the random Announcement Nonce from the IE.

Recipient may use the network key to decrypt and authenticate the IE, and after that verifying that the sequence number is larger than *priSequenceNumber* from *priNetworkDescriptor* then it knows that the announcement is not replay of announcement the device has seen before.

Message sequence chart to receive Net Announcement IE, and to request addresses is shown in Figure 11.

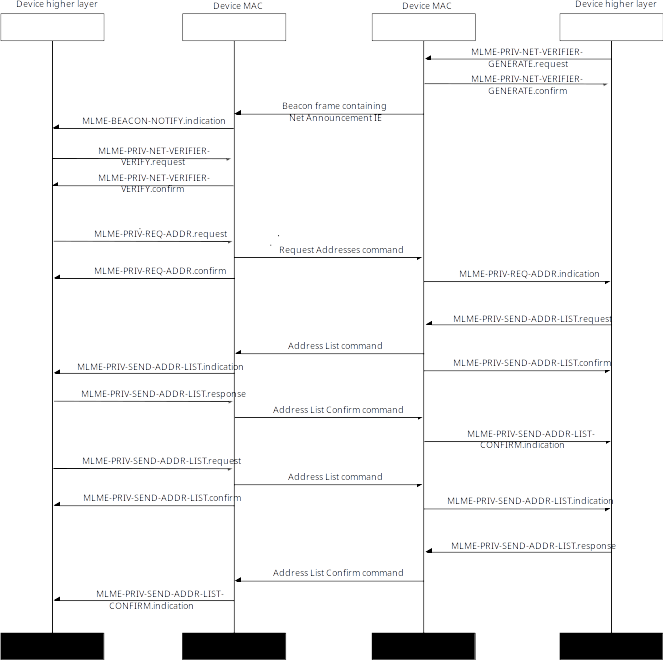
The owner of the network will generate encrypted verifier and includes that IE to the beacon frames. It does not need to generate new encrypted verifier for every single beacon, but it should generate new encrypted verifier and the incremented sequence number often enough, that if device who has been part of the network disconnects, and then reconnects it shall see new sequence number.

The device wanting to join the network will listen periodic announcements and when it hears the beacon it will use MLME-PRIV-NET-VERIFIER-VERIFY MLME primitive to find the network information associated with the IE. Then it can verify the sequence number and other information.

NOTE—Device does not need to listen network announcements all the time, it might also have knowledge that this specific network is for example only available in this physical location, and if it can use GPS or other source to find out when it is near that location, it may enable joining process only when it is close by to the network.

After the device has found out that the network is available in the area, it may request extended privacy addresses used by the network owner using MLME-PRIV-REQ-ADDR MLME primitive. The network owner responds with Address List command, and the joining device now knows the extended privacy addresses of the network owner. This exchange uses the source address of the network announcement frame as a destination address.

In the last step the joining device sends its own list of extended privacy addresses to the network owner.

Figure 11—Network announcement and exchanging privacy addresses

In case the device does not have a security keys for the network, after finding the network it wants to join, it may start IEEE Std 802.15.9 KMP with the network owner, and exchange the security keys needed.

* 1. Network request (Net Request IE)

Other method to find the network to join is to send request to the broadcast or multicast address and include Net Request IE in that frame. Doing this could be triggered by other means, for example when device detect it is home because it already joined home WiFi network, or using the GPS position of the device.

NOTE—If the network owner responds to the Net Request IE, then it is trivial to keep tracking of that, and mobile network owners should not respond to the Net Request IEs, but should instead use Network Announcement IE method to advertise network.

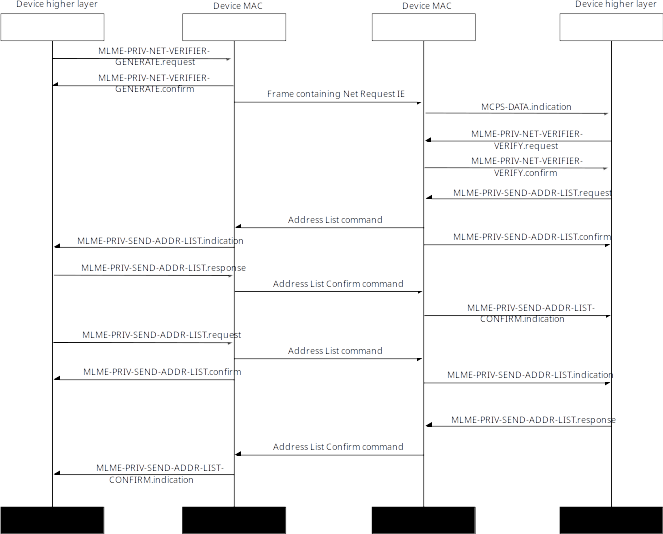
This request frame containing Net Request IE shall be sent using extended address as source address (most likely extended privacy address, and this extended address is used in nonce generation when encrypting and authenticating the Encrypted Request Verifier field). The destination address is typically multicast or broadcast address. The request should be sent without encryption so network owner can process it even when it does not have security keys to do security processing of the frame. The frame may be authenticated, as in case the network owner happen to have security context.

NOTE—Even if the frame is authenticated this does not protect against replays, as attacker can simply extract the Net Request IE from the valid frame it has sniffed from the valid joining device, and resend the frame without authentication, or with different key source, and the network owner would simply assume that the joining device and network owner does not have common security context anymore.

It may be sent encrypted in case device assumes the network owner recognizes source address, and can find security context based on that.

Processing is same as in the Net Announcement IE meaning if the recipient can verify the verifier, it can send Address List command to the sender of this message to update the addresses.

Message sequence chart to receive Net Request IE, and to request addresses is shown in Figure 12.

Figure 12—Network request IE use case example.

First the joining device will generate encrypted verifier and includes that IE to frame that is sent to the broadcast, multicast address, or directly to the network owner address.

The network owner receives this frame and it may then use MLME-PRIV-NET-VERIFIER-VERIFY MLME primitive to find the network information associated with the IE.

If it recognizes the device searching for the network, it may send Address List command to send updated extended privacy addresses to the joining device. The joining device may send its own list back.

In case the network owner does not have a security keys for the joining device, it may start IEEE Std 802.15.9 KMP with the joining device, and exchange the security keys needed.