Application Report
TIDA025–September 2018

Implementing a Pairwise Key Mechanism for the 6LoWPAN Mesh TI Designs

Wonsoo Kim

ABSTRACT

This article provides implementation details of how to extend the 6LoWPAN mesh TI design software examples with a pairwise private key mechanism via the TI-15.4 stack APIs. The pairwise key mechanism can be implemented to enhance security level, to make the software example compliant to a standard solution that requires the pairwise key mechanism, or both. The corresponding TI designs are TIDA-01547, TIDA-010003, and TIDA-010024.

Contents

Trademarks

1 TI-15.4 Security Overview

This section discusses the TI-15.4 frame format with the security header and the security PIBs, which follows the IEEE 802.15.4 standard.

1.1 TI-15.4 Stack Frame Format

The TI-15.4 stack follows the IEEE 802.15.4 frame format and the security mechanism. Figure 1 shows the IEEE 802.15.4 generic MAC frame format. If the MAC security is enabled with 1-bit Security Enabled flag in the Frame Control field, the auxiliary security header is conveyed with the MAC frame. The Auxiliary Security header consists of Security Control, Frame Control and Key Identifier sub-fields. The Security Control includes security level and key identifier mode. As shown in Table 1, the key identifier mode determines the number of octets of the following key identifier. The key identifier, combination of key source and key index, is the identifier to decrypt the received frames with the key information based on the identifier.

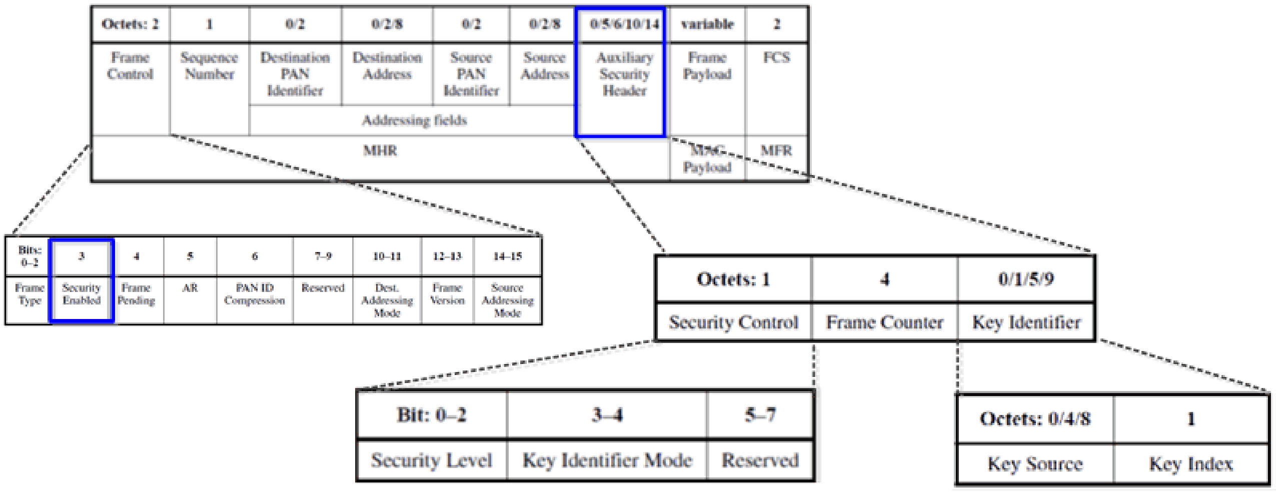

Figure 1. IEEE 802.15.4 Security Header Format

Table 1. IEEE 802.15.4 Key Identifier Mode

Key Identifier Mode	Key Identifier Field Length
0x00	0
0x01	1
0x02	5
0x03	9

1.2 Security MAC PIBs

Implementing a pairwise key mechanism requires to configure the key information associated with the corresponding devices via the macSecurityPIB. The macSecurityPIB is maintained by the TI-15.4 stack and the PIBs can be configured via the TI-15.4 stack API. There are three primary tables defined in the TI-15.4 stack to maintain the key information with the associated device information: MAC_KEY_TABLE, MAC_DEVICE_TABLE and MAC_SECURITY_LEVEL_TABLE.

The MAC_KEY_TABLE stores the security information for neighbor entries used for data encryption and decryption. Each entry in the key table contains key ID lookup data, key usage information and key along with the corresponding device information. The maximum number of entries, the size of the key table, is set via MAC_KEY_TABLE_ENTRIES at the initial stage. A way to add a new entry in the key table is to set the individual PIBs of the of MAC_KEY_ID_LOOKUP_ENTRY, MAC_KEY_USAGE_ENTRY, MAC_KEY_ENTRY and MAC_KEY_DEVICE_ENTRY that compose an entry in the key table. Section 2 discusses the implementation details based on this.

Table 2. MAC Key Table

PIB ID	Description
MAC_KEY_TABLE	A table of KeyDescriptor, entries, each containing keys and related information required for secured communications
MAC_KEY_TABLE_ENTRIES	The number of entries in macKeyTable
MAC_KEY_ID_LOOKUP_ENTRY	The key lookup table entry, part of an entry of the key table
MAC_KEY_USAGE_ENTRY	The key usage entry, part of an entry of the key table
MAC_KEY_ENTRY	The MAC key entry, an entry of the key table
MAC_KEY_DEVICE_ENTRY	The key device entry, part of an entry of the key table

The MAC_DEVICE_TABLE stores the neighbor device information. Each entry in the device table contains pan ID, device address (can be short address, extended address or both) and frame counter information. The MAC_DEVICE_TABLE_ENTRIES defines the maximum number of device entries in the table, that is, the size of the device table. This is typically configured at the initial stage. In the 6LoWPAN mesh TI design example, a new entry can be added to the TI-15.4 stack via the TI-15.4 stack API of Jdllc_addSecDevice.

Table 3. MAC Device Table

PIB ID	Description
MAC_DEVICE_TABLE	A table of Device-Descriptor entries, each indicating a remote device with which this device securely communicates
MAC_DEVICE_TABLE_ENTRIES	The number of entries in macDeviceTable
MAC_DEVICE_ENTRY	The MAC device entry, an entry of the device table

The MAC_SECURITY_LEVEL_TABLE maintains the security level descriptor entries and the MAC_SECURITY_LEVEL_ENTRY configures the maximum number of the table entries. The level descriptor defines the minimum level of security for incoming frames. Typically, this table is set at the initial stage and remains unchanged. In the TI design software example, this table is configured via the TI-15.4 stack API of Jdllc_securityInit at the initial stage.

Table 4. MAC Security Level Table

PIB ID	Description
MAC_SECURITY_LEVEL_TABLE	A table of SecurityLevel-Descriptor entries, each with information about the minimum security level expected depending on incoming frame type and subtype
MAC_SECURITY_LEVEL_TABLE_ENTRIES	The number of entries in macSecurityLevelTable
MAC_SECURITY_LEVEL_ENTRY	The MAC security level entry, an entry of the security level table

2 Implementing a Pairwise Key Mechanism

The pairwise key mechanism allows each link-level pair to negotiate a private key. With an example, we will discuss how a device can configure a private key with the given key identifier via the TI-15.4 stack APIs.

This example is based on TIDA-010003 and TIDA-010024, which implements the 6LoWPAN mesh stacks with the TI-15.4 stack in a single-chip of CC1310 or CC1312R. Therefore, all the APIs are internal. On the other hand, TIDA-01547 implements the 6LoWPAN mesh stacks in the external MSP432 MCU connected to the TI-15.4 co-processor in the CC1310 via UART. The same concept can be applied but with different set of APIs based on the host message protocol defined for the TI-15.4 co-processor. You can find the details of the host message protocol in the SimpleLink™ CC13x0 SDK.

2.1 Configuring a Pairwise Key via the TI-15.4 APIs

This example assumes that the security level is set to "ApiMac_secLevel_encMic32" and the security key ID mode is set to "ApiMac_keyIdMode_8", and thus the length of key identifier is 9B. The 1B key index is set to 0x02 and the 9B key identifier (or lookup data) is constructed by combining 8B default key source (same for all the devices in this example) and 1B key index. The 9B lookup data is unique because of the unique 1B key index value.

The codes below show an example to configure a private key via the TI-15.4 stack APIs. To configure a new private key, a device needs to configure the Security PIBs of MAC_KEY_ID_LOOKUP_ENTRY, MAC_KEY_USAGE_ENTRY and MAC_KEY_ENTRY with a unique key index (0x02 in this example) to create a new entry in the key table. The macWrapperAddDevice is an API to configure MAC_KEY_DEVICE_ENTRY in the key table and to link the device in the MAC device table. The corresponding sender and receiver should configure the same security information except for the device information. Once the configuration is completed, the new entry with the security information will be stored in the key table with the new array index of the key index value (0x02). The complete source codes can be found in the TIDA-010024 TI design example that implements a MAC-level DTLS mechanism.

```
macSecurityPibKeyUsageEntry_t macKeyUsageEntry;
macSecurityPibKeyEntry_t macKeyEntry;
macSecurityPibKeyIdLookupEntry_t macKeyIdLookupEntry;
uint8_t status=ApiMac_status_unsupportedType;
uint8_t new_key_index=0x02;

/* Add Lookup entry in the key table*/
macKeyIdLookupEntry.key_index = new_key_index;
macKeyIdLookupEntry.key_id_lookup_index = 0;
// unique lookup data with the new key index
memcpy(macKeyIdLookupEntry.macKeyIdLookupEntry.lookupData,
keyIdLookupList[0].lookupData,(APIMAC_KEY_SOURCE_MAX_LEN));
macKeyIdLookupEntry.macKeyIdLookupEntry.lookupData[APIMAC_KEY_SOURCE_MAX_LEN]= new_key_index;
macKeyIdLookupEntry.macKeyIdLookupEntry.lookupDataSize = 0x01;

status=MAC_MlmeSetSecurityReq(MAC_KEY_ID_LOOKUP_ENTRY, &macKeyIdLookupEntry);

/* Add keyUsage entry in the key table */
macKeyUsageEntry.key_index= new_key_index;
macKeyUsageEntry.key_key_usage_index=0;
memcpy(&macKeyUsageEntry.macKeyUsageEntry, keyUsageList, sizeof(keyUsageList));
status=MAC_MlmeSetSecurityReq(MAC_KEY_USAGE_ENTRY, &macKeyUsageEntry);

/* Add key entry */
macKeyEntry.key_index = new_key_index;
macKeyEntry.frameCounter=0;
memcpy(macKeyEntry.keyEntry, key, sizeof(macKeyEntry.keyEntry));
status=MAC_MlmeSetSecurityReq(MAC_KEY_ENTRY, &macKeyEntry);

/* Add key device entry */
status=macAddDevice(dtlsDevTable[index].devAddr.addr, new_key_index);
```

2.2 Data Transfer With a New Private Key

The next step is to transfer data with the new key. The code below shows how to fill the security header in the data frame with the new key index value of 0x02. It is recommended to configure the security header information via "Jdllc_securityFill" which is an API provided by the TI-15.4 stack. As this example assumes the same key source, security level and key ID mode, the key index is the only value to be updated.

At the receiver side, there are no additional codes to handle the new private key. The decryption of the received data can be done in the TI-15.4 stack based on the new key configuration discussed in Section 2.1. The application, running on top of the TI-15.4 stack, will receive decrypted data via the registered data indication callback.

```
void Jdllc_securityFill(ApiMac_sec_t *pSec, uint8_t *pDstAddr)
{
    // default pre-shared key
    pSec->securityLevel = secLevel;
    pSec->keyIdMode = secKeyIdMode;
    memcpy(pSec->keySource,keyIdLookupList[0].lookupData,(APIMAC_KEY_SOURCE_MAX_LEN));
    pSec->keyIndex = 0x02;
}
```

3 References

1. *Simple 6LoWPAN Mesh Data Collector Improves Network Performance Reference Design* (TIDA-01547)
2. *Simple 6LoWPAN Mesh End-Node Improves Network Performance Reference Design* (TIDA-010003)
3. *Secured 6LoWPAN Mesh End-Node with Enhanced Network Capacity Reference Design* (TIDA-010024)
4. ieee.org, *802.15.4-2011 - IEEE Standard for Local and metropolitan area networks--Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs), 2011*