Telink BLE Mesh Lighting APP Specification

PS-BLE-15071300-E4

Ver 1.3.0

2015/7/21

Brief：

This document is a specification for Telink BLE (Bluetooth Low Energy) Mesh Lighting APP.

TELINKSEMICONDUCTOR

Published by
Telink Semiconductor

Bldg 3, 1500 Zuchongzhi Rd,
Zhangjiang Hi-Tech Park, Shanghai, China

Legal Disclaimer

Telink Semiconductor reserves the right to make changes without further notice to any products herein to improve reliability, function or design. Telink Semiconductor disclaims any and all liability for any errors, inaccuracies or incompleteness contained herein or in any other disclosure relating to any product.

Telink Semiconductor does not assume any liability arising out of the application or use of any product or circuit described herein; neither does it convey any license under its patent rights, nor the rights of others

The products shown herein are not designed for use in medical, life-saving, or life-sustaining applications. Customers using or selling Telink Semiconductor products not expressly indicated for use in such applications do so entirely at their own risk and agree to fully indemnify Telink Semiconductor for any damages arising or resulting from such use or sale.

Information:

For further information on the technology, product and business term, please contact Telink Semiconductor Company (www.telink-semi.com).

For sales or technical support, please send email to the address of:

telinkcnsales@telink-semi.com

telinkcnsupport@telink-semi.com

Change Log

Version	Main Changes	Date	Authors
1.0.0	Initial Release	2015/7	J.H.P., ZMJ, JGH, Cynthia
1.1.0	Added Mesh Online status update; Added description for opcode “0xdc”.	2015/7	J.H.P., ZMJ, Cynthia
1.2.0	Added opcode “0xE2”	2015/7	JGH, Cynthia
1.3.0	Added spec on Mobile App Cloud requirements	2015/7	JHP, Cynthia

Table of Content

Table of Figures

Table of Tables

1 Introduction

This document defines the Mobile App functions required to control the Telink BLE mesh networks and devices. The interface between the Mobile App and the BLE light nodes is described in details.

2 APP Function

2.1 Mesh form and control

1) List all factory default light/switch as candidate for adding to the mesh network:
   a. Scan QR code to obtain light/switch info if needed
   b. Add light/switch to one or more groups (e.g. bedroom, living room, …)
2) Allow the user to select factory-default light and switch to start a mesh network.
3) Allow the user to select a factory-default light and switch and add to an existing mesh network.
4) Allow to remove any light from the mesh network.

2.2 Mesh provision

1) When starting mesh or adding new nodes into the mesh, APP can provision and store the following information:
   a) Network name (human configurable and readable);
   b) Password (human readable and configurable);
   c) Network long term key (not human readable, APP generates a random number of required length).
2) APP may change the light/switch information:
   a) Light/switch name;
   b) The group(s) a light/switch may be assigned to (Group name is presented as human readable strings such as “living room” in the UI, but the actual Group ID are two-byte integers), one light node may belong to one or more groups.

2.3 Control of light or group of lights

1) APP displays the status of all the lights within the mesh network (online/offline, on/off, color RGB, color temp, brightness);
2) APP allows one light or a group of lights to be selected and controlled for on/off, RGB, color temp, and brightness.

2.4 Obtain network key

1) When the user already has one mesh network setup in the home, APP can scan and discover devices in the mesh network, connect to the device with correct network name and password (obtained through OOB method such as human communication or cloud sharing), and obtain the network long term key (LTK).

2.5 OTA Firmware Upgrade

1) Mobile APP can pick a binary file, choose one BLE light, and perform firmware upgrade over the air for the BLE light.
   a. The binary file can be downloaded to the Mobile APP by scanning a QR code to obtain the URL.

2.6 App interaction with Cloud

1) The App provides a login interface to the cloud. After the user logged in from the UI, the user's profile information can be obtained from the Cloud.
   a. Details of user profile information are defined by the Cloud provider.
2) The App stores mesh related information on to the Cloud.
   a. The detailed mesh information to be stored include:
      i. Mesh network name
      ii. Mesh network password
      iii. Mesh network LTK
      iv. All group IDs and corresponding group names
      v. For each light/switch device, store the following information
         1. ManufactureID
         2. MeshUUID
         3. Device MAC
         4. Device Mesh Address
         5. ProductUUID
         6. List of group IDs for the groups that the light belongs to.

b. There needs to be a mechanism for future change or extension of the above Mesh network information block. (e.g. by using a mesh information block version number to identify format change)

3 Communication Protocol between Mobile App and the BLE Mesh Lights

3.1 Introduction

In Telink BLE Mesh network, the Mobile App can discover any light or switch within the mesh network, establish connections, and exchange control command or status information. As shown in Figure 1, the connection between Mobile App and the connected light is a 1-to-1 normal BLE connection; within the mesh network, the control and status information are exchanged with N-to-M relationship.

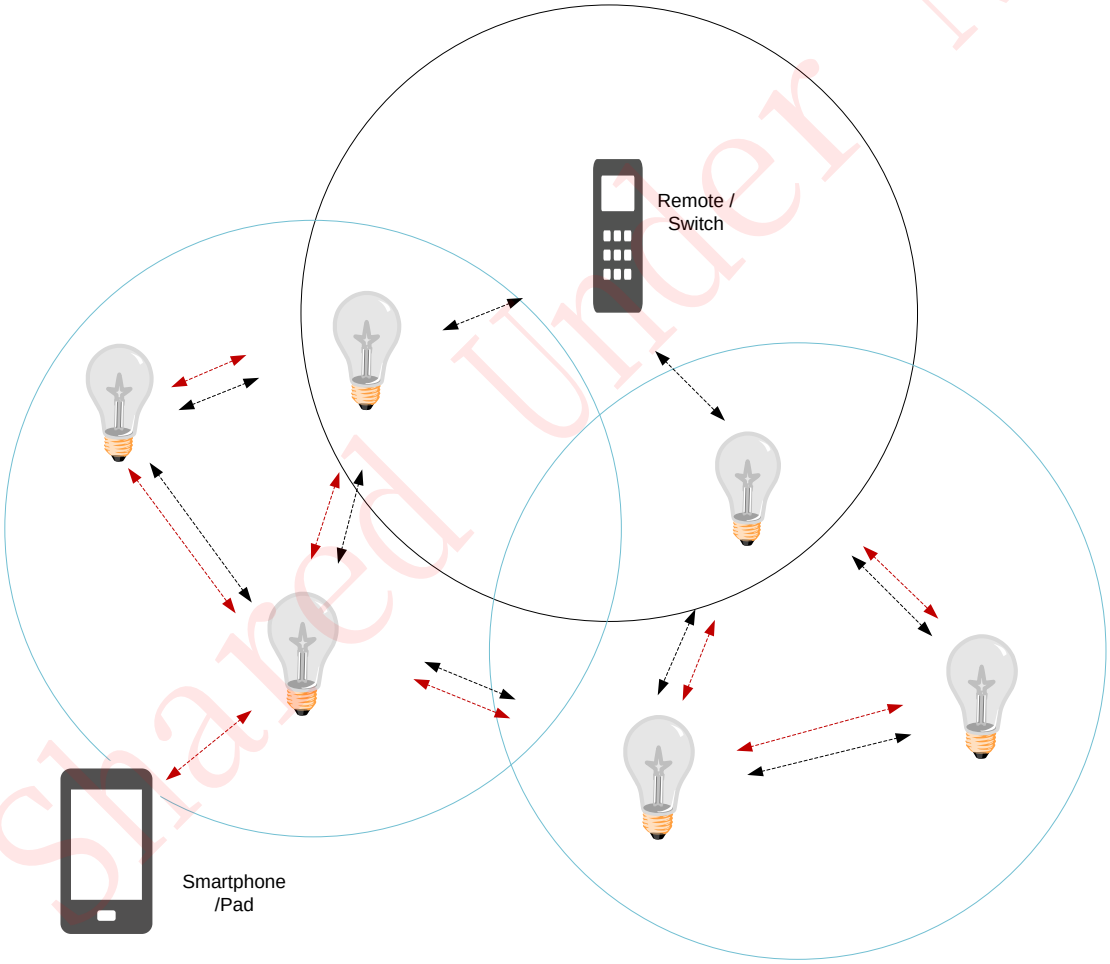

Figure 1 BLE Mesh Light Control Scenarios

From the Mobile App point of view, it is only communicating with one BLE light at any time but can control other nodes in the network or obtain information about

other nodes through this connected node.

The general communication flow between Mobile App and the BLE mesh light is summarized as follows:

1. BLE light periodically broadcasts Adv packets on BLE advertising channel;
2. Mobile App scans for the Adv packets and establishes connections with the BLE light;
3. Depending on the Mobile App and BLE light state, the following cases can happen:
    a. If Mobile App does not control a mesh and the BLE light is factory default, the Mobile App needs to create and provide network config information to the BLE light;
    b. If Mobile App controls a mesh and the BLE light is factory default state, then the Mobile App needs to provide the BLE light with the existing network config information;
    c. If Mobile App does not control the mesh and the BLE light belongs to a mesh, then the Mobile App needs to pass basic password authentication and then obtain the network configuration information from the BLE light;
    d. If Mobile App and the BLE light are already configured to the same mesh network, then they simply establish connection using existing network config information.
4. Once secure (or optionally unsecured) connection is established between the Mobile App and the BLE light, the Mobile App can issue command to any node or group within the mesh network through this connection.

The remaining part of this section describes in details the protocol for achieving the above steps.

3.2 Device Advertising

The BLE light devices sends periodic ADV_IND type advertising packet to broadcast mesh and light related information. Other than the AdvA (advertiser address) field, the advertising packet contains the data shown in Table 1.

Table 1 Advertising data

Length	Type	Data	Description
02	01	05	Device connectable
0D	09	74 65 6C 69 6E 6B 5F 6D 65 73 68 31	MESH_NAME: telink_mesh1
09	FF	11 02 11 02 ab cd ef gh	Private data definition BYTE[1:0]: ManufactureID BYTE[3:2]: MeshUUID BYTE[7:4]: Device MAC address

*Note：Type 01/09/FF are standard BLE definition values.

MESH_NAME: Human readable mesh name, up to 16 characters.

ManufactureID: Numerical identifier for different light vendors such as GE, OSRAM, Philips, …

MeshUUID: Mesh product identifier, used to distinguish major feature differences in Telink mesh solution, e.g., full mesh, reduced function mesh, …

Device MAC address: lower 4 bytes of the device MAC

A captured Adv packet is shown below in Figure 2.

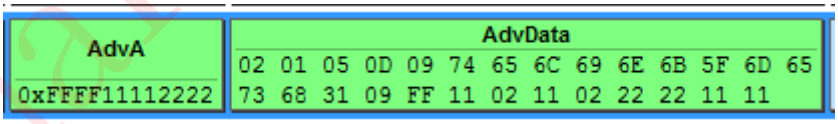

Figure 2 Captured Adv packet example

3.3 Scan for Device Info

The Mobile App may scan for BLE light information without establishing connections to the BLE light. To get the light device information, the Mobile App sends BLE standard SCAN_REQ after receiving the ADV_IND packet from the BLE light.

The BLE light responds with SCAN_RSP to provide device information to the requesting Mobile App. The SCAN_RSP takes the standard format and contains the private data defined in Table 2.

Table 2 Scan response data

Length	Type	Data	Description
0C	FF	11 02 11 02 ab cd ef gh 00 00 00	Private data definition BYTE[1:0]: ManufactureID BYTE[3:2]: MeshUUID BYTE[7:4]: Device MAC address BYTE[9:8]: ProductUUID BYTE[10]: Status BYTE[12:11]: DeviceAddress

ProductUUID: Numerical identifier for product type, such as bulb, switch, …

Status: It will be increased by one to indicate status change.

DeviceAddress: 16-bit device address in mesh network

A SCAN_RSP packet example is shown in Figure 3.

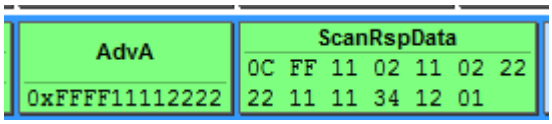

Figure 3 Captured scan response packet example

3.4 Connection Establishment and Service Discovery

Once the Mobile App discovers the BLE light through the advertising or scan described in Section 3.2 and Section 3.3, it can establish connection to the BLE light by sending a standard BLE CONNECT_REQ packet as shown in Figure 4.

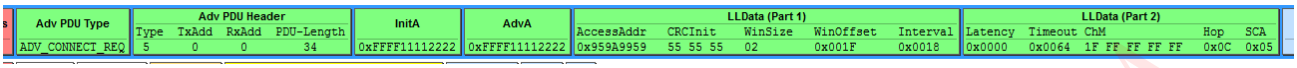

Figure 4 Connection Request Example

The connection is established between the Mobile APP and the BLE light following the standard BLE procedure. On top of the connection, the Mobile APP will follow the standard BLE procedures to discovery all services supported by the BLE light.

Table 3 lists the BLE attributes used for mesh services. A customized service with UUID 0x00010203040506070809 0A0B0C0D1910 is defined as the Mesh Light Access Service. The following characteristics are defined within the service for Mesh light operation:

- Mesh light status: used for light status query
- Mesh light command: used for light control command
- Mesh light OTA: used for over-the-air firmware update
- Mesh light pair: used for pairing, provision, and security setup

Table 3 Mesh Attributes

Type (hex)	Type	Hex/Text Value	GATT Server Permissions	Notes
0x2800	GATT Primary Service	0x1800(Generic Access Service)	GATT_PERMIT_READ	Generic Access Service
0x2803	GATT Characteristic	02(properties:Read) 03 00(handle:0x0003) 00 2A(UUID:0x2A00)	GATT_PERMIT_READ	Device Name characteristic declaration
0x2A00	Device Name		GATT_PERMIT_READ	Device Name characteristic value
0x2803	GATT Characteristic	02(properties:Read) 05 00(handle:0x0005)	GATT_PERMIT_READ	Appearance characteristic

		012A(UUID:0x2A01)		declaration
0x2A01	Appearance		GATT_PERMIT_READ	Appearance characteristic value
0x2803	GATT Characteristic	02(properties:Read) 07 00(handle:0x0007) 04 2A(UUID:0x2A04)	GATT_PERMIT_READ	Peripheral Preferred Connection Parameters characteristic declaration
0x2A04	Peripheral Preferred Connection Parameters		GATT_PERMIT_READ	Peripheral Preferred Connection Parameters characteristic value
0x2803	GATT Characteristic	02(properties:Read) 09 00(handle:0x0009) 262A(UUID:0x2A26)	GATT_PERMIT_READ	Firmware Revision String characteristic declaration
0x2A26	Firmware Revision String		GATT_PERMIT_READ	Firmware Revision String characteristic value
0x2803	GATT Characteristic	02(properties:Read) 0B 00(handle:0x000B) 292A(UUID:0x2A29)	GATT_PERMIT_READ	Manufacturer Name String characteristic declaration
0x2A29	Manufacturer Name String		GATT_PERMIT_READ	Manufacturer Name String characteristic value
0x2803	GATT Characteristic	02(properties:Read) 0D 00(handle:0x000D) 242A(UUID:0x2A24)	GATT_PERMIT_READ	Model Number String characteristic declaration
0x2A24	Model Number String		GATT_PERMIT_READ	Model Number String characteristic value
0x2803	GATT Characteristic	02(properties:Read) 0F 00(handle:0x000F) 272A(UUID:0x2A27)	GATT_PERMIT_READ	Hardware Revision String characteristic declaration
0x2A27	Hardware Revision String		GATT_PERMIT_READ	Hardware Revision String characteristic value
0x2800	GATT Primary	0x00010203040506070 80	GATT_PERMIT_READ	Mesh LIght Access

	Service	90A0B0C0D1910(Mesh LIght Access Service)		Service
0x2803	GATT Characteristic	12(properties:Read/Notify) 12 00(handle:0x0012) 11190D0C0B0A09080706050403020100(UUID: 0x000102030405060708090A0B0C0D1911)	GATT_PERMIT_READ	Mesh Light Status characteristic declaration
0x0001020304050607080 90A0B0C0D1911	Mesh Light Status		GATT_PERMIT_READ	Mesh Light Status characteristic value
0x2803	GATT Characteristic	0A(properties:Read/Write) 15 00(handle:0x0015) 12190D0C0B0A09080706050403020100(UUID: 0x000102030405060708090A0B0C0D1912)	GATT_PERMIT_READ	Mesh Light Command characteristic declaration
0x0001020304050607080 90A0B0C0D1912	Mesh Light Command		GATT_PERMIT_READ	Mesh Light Command characteristic value
0x2803	GATT Characteristic	0A(properties:Read/Write_Without_Rsp) 18 00(handle:0x0018) 13190D0C0B0A09080706050403020100(UUID: 0x000102030405060708090A0B0C0D1913)	GATT_PERMIT_READ	Mesh Light OTA characteristic declaration
0x0001020304050607080 90A0B0C0D1913	Mesh Light OTA		GATT_PERMIT_READ	Mesh Light OTA characteristic value
0x2803	GATT Characteristic	0A(properties:Read/Write_Without_Rsp)	GATT_PERMIT_READ	Mesh Light Pair characteristic

		1B 00(handle:0x001B) 14190D0C0B0A09080706 050403020100(UUID: 0x0001020304050607080 90A0B0C0D1914)		declaration
0x0001 020304 050607 08090A 0B0C0D 1914	Mesh Light Pair		GATT_PERMIT_READ	Mesh Light Pair characteristic value

3.5 Pairing, Provision, and Security Setup

The pairing process between the Mobile App and the BLE light is performed using the Pair attribute with the command format and opcode defined in Table 4 and Table 5.

Table 4 Pairing command format

Opcode (1 byte)	Parameters (0 – 16 bytes)

Table 5 Opcode definition

Operation	Code	Parameter
BLE_GATT_OP_PAIR_REQ	1	None
BLE_GATT_OP_PAIR_RSP	2	None
BLE_GATT_OP_PAIR_REJECT	3	None
BLE_GATT_OP_PAIR_NETWORK_NAME	4	16-byte: ER(SK, network_name)
BLE_GATT_OP_PAIR_PASS	5	16-byte: ER(SK, password)
BLE_GATT_OP_PAIR_LTK	6	16-byte: ER(SK, LTK)
BLE_GATT_OP_PAIR_CONFIRM	7	None
BLE_GATT_OP_PAIR_LTK_REQ	8	Byte0~7: random number (rm) Byte8~15: low 8 bytes of ER(rm, network_name ^ password)
BLE_GATT_OP_PAIR_LTK_RSP	9	16-byte: ER(skr, ltk) skr= rm^network_name^password
BLE_GATT_OP_PAIR_DELETE	10	Byte0~7: random number (rm) Byte8~15: low 8 bytes of ER(rm, network_name ^ password)
BLE_GATT_OP_PAIR_DEL_RSP	11	No

BLE_GATT_OP_ENC_REQ	12	Byte0~7: random number (rm) Byte8~15: low 8 bytes of ER(rm, network_name ^ password)
BLE_GATT_OP_ENC_RSP	13	Byte0~7: random number (rs) Byte8~15: low 8 bytes of ER(rs, network_name ^ password)
BLE_GATT_OP_ENC_FAIL	14	No

Session Key (SK) generation
SK = ER (network_name ^ password, rm|rs)
*Note: The “ER” function used in the above format description the same as the one defined in Bluetooth Specification for AES-128.

As described in Section 3.1, depending on the phone and the BLE light node status, there are several scenarios when the phone and the BLE light node establish connections. The rest of this section describes what type of information is exchanged in each case and how the security key is setup.

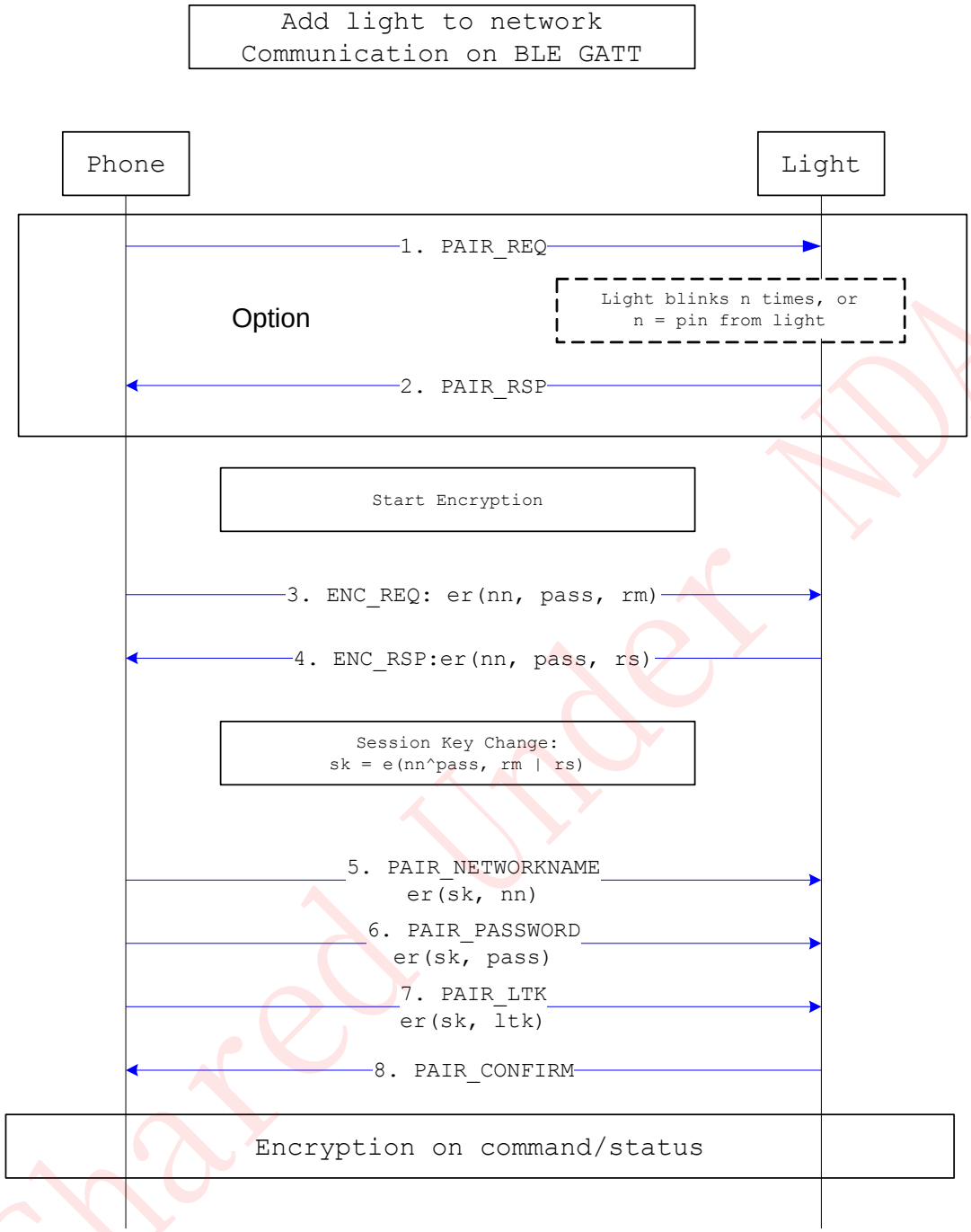

Figure 5 Form new mesh or add new phone

The steps in Figure 5 show how to form a new mesh or how a new phone is added into an existing mesh network. During this process, the new mesh node is configured with (network name, password, and ltk) information. The network name and password are human readable and generated by the end user. The “ltk” is generated by the Mobile App automatically as a long term key to protect communication within

the mesh network and is not visible through user interface.

The actions in each step are described below:

(Optional) Step 1~2, PAIR_REQ from the Mobile App to the mesh node. This request will trigger the light node to provide some visual feedbacks (e.g. blink several times) which help the end user to confirm that the selected light is the one to be added into the mesh network. The BLE light node sends PAIR_RSP to confirm the reception of PAIR_REQ.

Step 3~4, ENC_REQ and ENC_RSP is exchanged between the Mobile App and the mesh node. During this exchange, the Mobile App and the mesh node establish a random pair (rm, rs), which will be used to derive session key for application level authentication and encryption during the connection. The session key is derived as sk = er(nn^pass, rm|rs).

Step 5 ~ 8, the Mobile App sends network name, password, mesh LTK to the mesh node and get confirmation, all information are protected by the session key. The mesh node will store (network name – nn, password – pass, long term key – ltk) as the mesh config information for mesh network communication usage.

After the above steps, the Mobile App and the mesh node are considered successfully paired and they can exchange information using the format described in Section 3.6. The Mobile App can terminate the connection with the mesh node any time after all the needed control commands are executed.

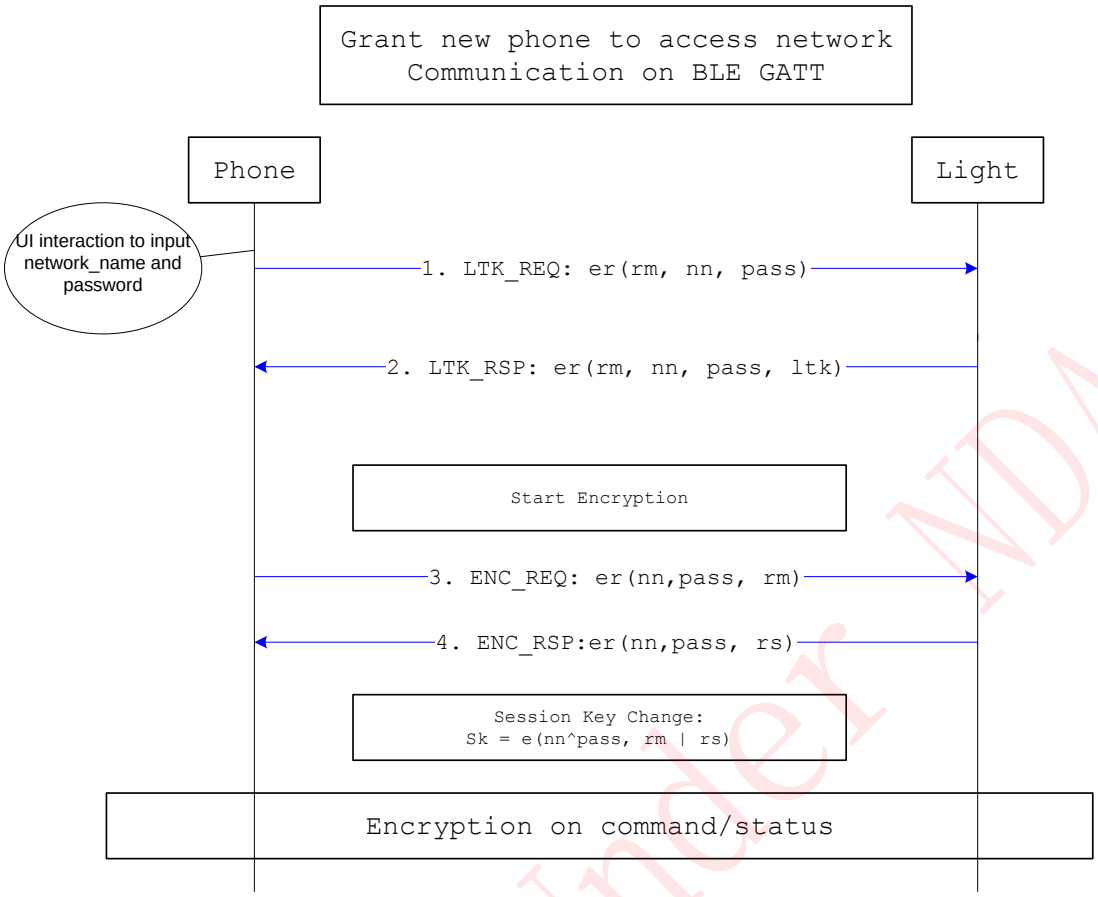

Figure 6 Adding new controlling phone into mesh network

The steps in Figure 6 show how to add a new phone into an existing mesh network. This is achieved by the Mobile App on the new phone communicating with any BLE light nodes in the mesh network. During this process, the new phone obtains the tuple (network name, password, ltk) through a combination of out-of-band and in-band methods.

The actions in each step are described below:

Step 0, The user enters the network name and password information through user interface prompt. This info is obtained using out of band method.

Step 1 ~ 2, The Mobile App and the mesh node exchanges LTK_REQ and LTK_REP packets. The ltk is passed by the mesh node to the Mobile App and the packet is protected with a temporary key generated based a random number rm from the Mobile App, the network name, and the password. After these steps, the Mobile

App/Phone is configured fully to access any other nodes within the mesh network.

Step 3 ~ 4, The Mobile App and the mesh node exchanges ENC_REQ and ENC_RSP to establish (rm, rs) pairs for session key derivation. Once session key is derived, the Mobile App can start issue commands to the mesh network though the connected mesh nodes. All communications are protected by the session key (sk).

The Mobile App can terminate the BLE connection with the mesh node any time after all the needed control commands are executed.

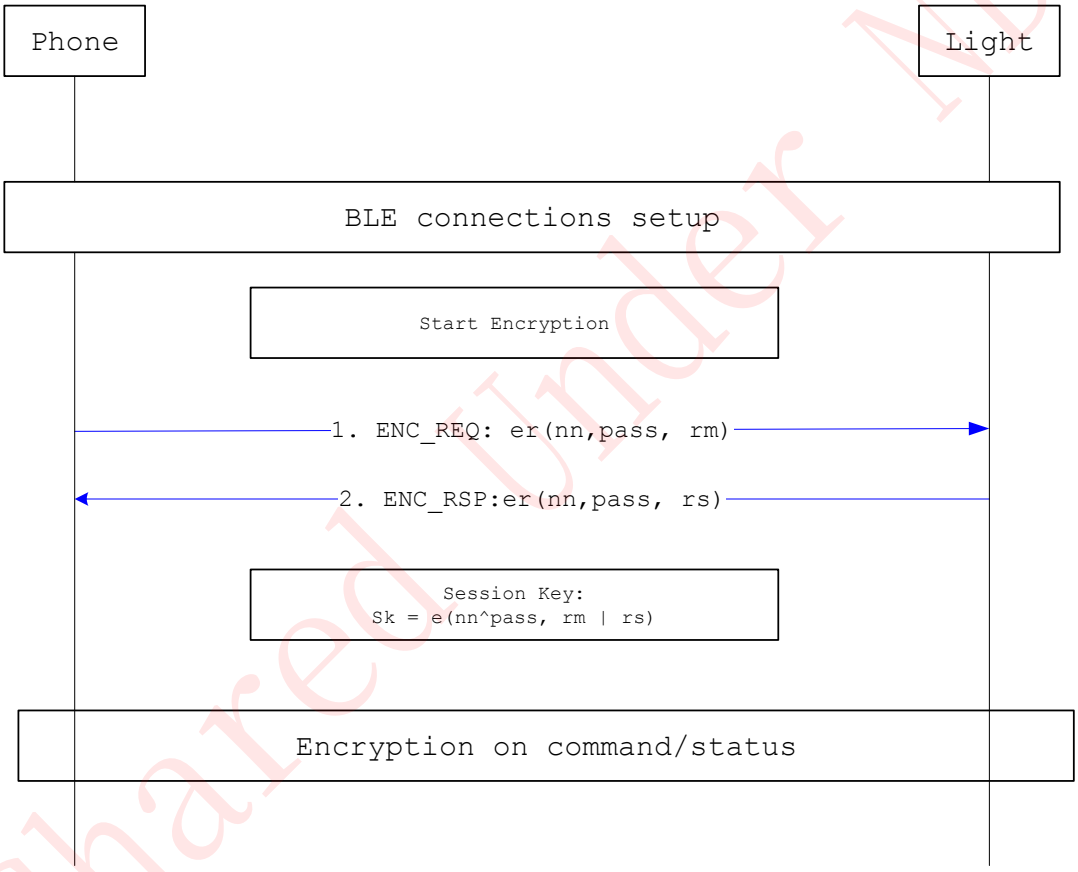

Figure 7 Normal phone/light connection (Both have network config info)

The steps in Figure 7 show existing phones and mesh nodes in the mesh network establishes security relationship for mesh controls. Both sides have the tuple (network name, password, ltk) already. So the steps are very simple:

Step 1 ~ 2, once the BLE connection is established, the Mobile App and the mesh node exchanges ENC_REQ and ENC_RSP to establish (rm, rs) pairs for session key

derivation. Once session key is derived, the Mobile App can start issue commands to the mesh network though the connected mesh nodes. All communications are protected by the Session Key (sk).

The Mobile App can terminate the BLE connection with the mesh node any time after all the needed control commands are executed.

3.6 Mesh Control

Once the connection has been established and the application level encryption is started as described in Section 3.5, the Mobile App and the BLE light can exchange commands to control any nodes within the network or obtain status of any nodes.

The packet formats used between the Mobile App and the BLE light node are shown in Figure 8 and Figure 9.

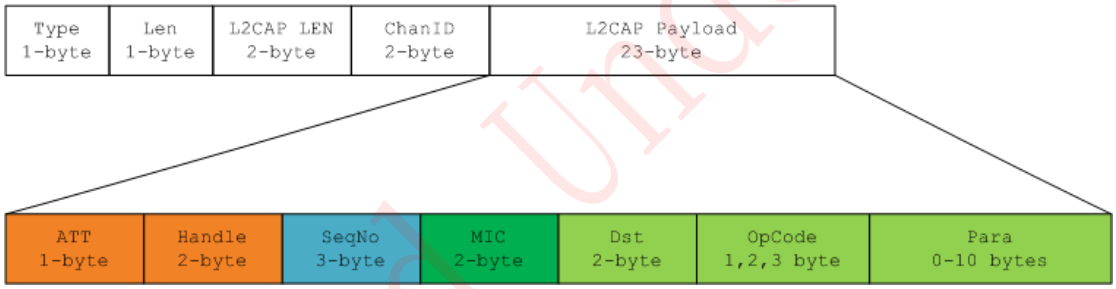

Figure 8 Packet format from Phone to Mesh Node (Light/Switch/...)

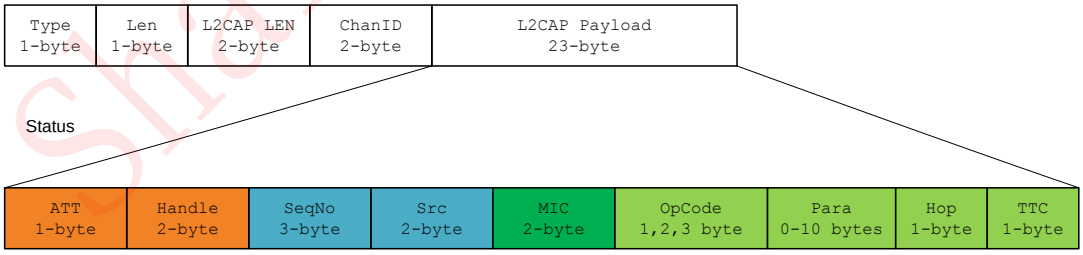

Figure 9 Packet format from Mesh node to Phone

The application level PDU (L2CAP Payload) formats used in the two directions are slightly different. However, in both case, application level authentication and encryption are used to protect the content exchanged. In both formats, fields in

"Orange" and "Blue" are sent as clear text; fields in "Blue", "Dark Green" and "Green" are covered by the authentication MIC; fields in "Green" are covered by the encryption.

All the authentication and encryption are calculated use the session key (SK) established in Section 3.5. The IV (Nonce) used in the authentication and encryption are different on the two directions. The format is given below in Table 6 and Table 7.

Table 6 IV formation (Phone to mesh nodes)

IV (Nonce)	Descriptor
Byte0~3	Mesh Device MAC address, low 4 bytes
Byte4	FF
Byte5~7	SeqNo: Command sequence number

Table 7 IV formation (Mesh nodes to phone)

IV (Nonce)	Descriptor
Byte0~2	Mesh Device MAC address, low 3 bytes
Byte3~5	SeqNo: Command sequence number
Byte6~7	Source address

For both the authentication and encryption, multiple choices of low level algorithms can be used, for example, AES or polynomial based algorithms. The choice of algorithm is identified by the MeshUUID in the advertising packet. Sample code AES based authentication and encryption is given in Section 4.1.

3.6.1 Mesh Opcodes and Parameters

As shown in Figure 8 and Figure 9, all the commands to and from the mesh network contain 1~3 bytes of Opcode and 0~10 bytes of parameters. The currently defined opcode and parameters are specified in Table 8, only 3-byte opcode is used for now and the opcode takes the format of 11xxxxxxzzzzzzzzzzzzzzzz，where "zzzzzzzz" is the two byte company id(0x0211 for Telink). All opcodes and parameters in Table 8 use handle of the Mesh light command except for 0xD4/D5/D6/DB, which use the handle corresponding to the Mesh light status attribute, both are defined in Section 3.4.

Table 8 Mesh Opcodes and parameters

OP	xxxxxx	Description (length and meaning of bytes in params field)		
0xD0	010000	Params[0]=1, light on		
		Params[0]=0, light off		
0xD1	010001	Reserved		
0xD2	010010	Set brightness and color, Params[0~5] corresponds to 6 channel PWM, the meaning of each channel is user configurable in the light node, for example, R/G/B/brightness/color. For each byte, PMW_MAX_TICK_MIN is the lowest value and 0xFF is the max value		
0xD3	010011	Reserved		
0xD4	010100	Response to short group ID query, Params[0-7] is the lower bytes of the 8 groups the light node belongs to, 0xFF indicates a null group		Each light node can belong to 8 groups at the most.
0xD5	010101	Response to long group ID query,Params[0-7] is the group ID for the first four groups the light node belongs to, 0xFFFF indicates a null group		
0xD6	010110	Response to long group ID query, Params[0-7] is the group ID for the last four groups the light node belongs to, 0xFFFF indicates a null group		
0xD7	010111	Params[0]=1, add group	Params[1~2], group id	
		Params[0]=0, remove group		
0xD8	011000	Reserved		
0xD9	011001	Reserved		
0xDA	011010	Status query， Params[0]: relay count		
0xDB	011011	Status response，Params[0~5] the PWM setting of the light node,		

		Params[6-7] Reserved, Params[8]=TTC, Params[9]=hops		
0xDC	011100	Online status report: Param[0~1]: device address for node0; 0xffff indicates null device Param[2]: 0 for offline; otherwise online Param[3]: device status for node0 Param[4~5]:device address for node1; 0xffff indicates null device Params[6]: 0 for offline; 1 for online Params[7]: device status for node1		
0xDD	011101	Group ID query, Params[0] is the relay count	Params[1]=1, query for short group ID (the lower bytes of the group IDs for up to 8 gorups)	
			Params[1]=2, query for long group ID of the first four groups	
			Params[1]=3, query for long group ID of the last four groups	
0xDE	011110	Reserved		
0xDF	011111	Reserved		
0xE0	100000	Set device address,Params[0-1]is the new device address.		
0xE1	100001	Notify the device address information,Params[0~1]is the device address.		
0xE2	100010	configure RGB's value respectively, params[0]: 1=R 2=G 3=B parmas[1]: luminance from 0% to 100%		

3.6.2 Mesh Online status update

Once APP writes 1 to light status characteristic, the online status of each node in mesh will be notified to the APP, and any status changes later on will also be updated to APP.

3.7 OTA

The OTA firmware update is achieve by using the customized UUID, with ATT data format shown in Table 9.

Table 9 OTA packet format

dat[23]	Description
0~1	Serial number (starting from 0x0000)
2~17	16-byte data of the new bin file
18~19	crcvalue of previous 18 bytes

Some sample OTA packets are shown in Figure 10. Each packet contains 20Bytes payload except for the last one.

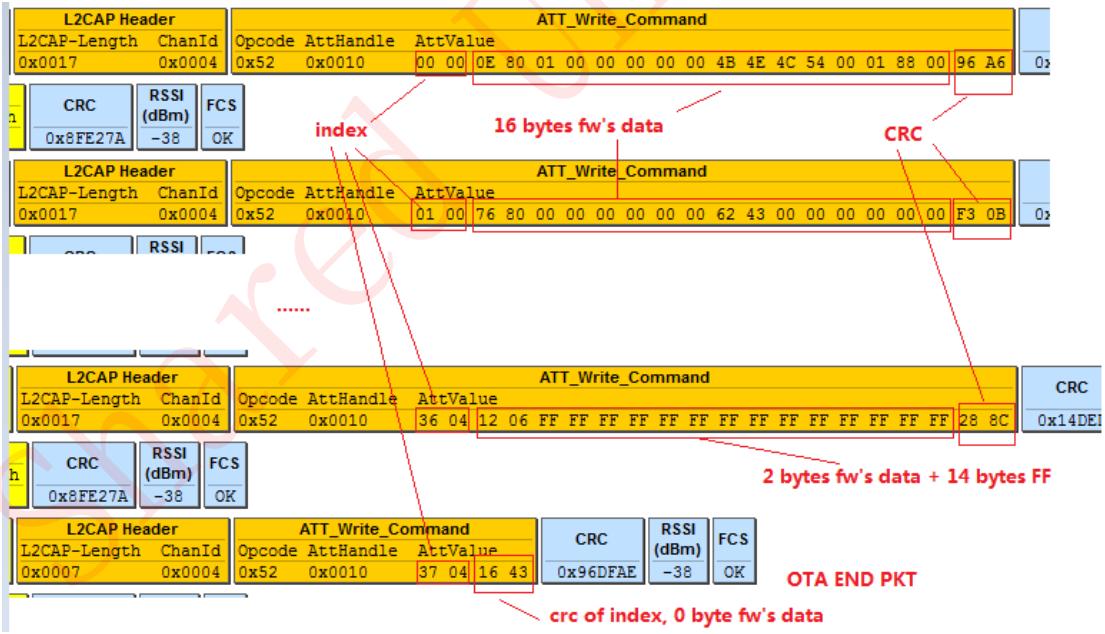

Figure 10 Sample OTA packet

4 Appendix

4.1 Sample code for AES based authentication and encryption

The function "ER" and "aes_att_encryption" are the same as defined in BLE standard for AES-128 function.

```
output_encrypted_text  = aes_att_encryption(key, input_plain_text)
```

Packet encryption and decryption use AES-CCM algorithm.

Packet encryption

```
u8 aes_att_encryption_packet(u8 *key, u8 *iv, u8 *mic, u8 mic_len, u8 *ps, u8 len)
{
    u8  e[16], r[16];

    ////////////// calculate mic ///////////////////////
    memset (r, 0, 16);
    memcpy (r, iv, 8);
    r[8] = len;
    aes_att_encryption (key, r, r);

    for (int i=0; i<len; i++)
    {
        r[i & 15] ^= ps[i];

        if ((i&15) == 15 || i == len - 1)
        {
            aes_att_encryption (key, r, r);
        }
    }
    for (int i=0; i<mic_len; i++)
    {
        mic[i] = r[i];
    }

    ////////////////// calculate enc ////////////////////////
    memset (r, 0, 16);
    memcpy (r+1, iv, 8);
    for (int i=0; i<len; i++)
    {
        if ((i&15) == 0)
        {
```

```
            aes_att_encryption (key, r, e);
            r[0]++;
        }
        ps[i] ^= e[i & 15];
    }

    return 1;
}
```

Packet decryption

```
u8 aes_att_decryption_packet(u8 *key, u8 *iv, u8 *mic, u8 mic_len, u8 *ps, u8 len)
{
    u8   e[16], r[16];

    ////////////////// calculate enc ////////////////////////
    memset (r, 0, 16);
    memcpy (r+1, iv, 8);
    for (int i=0; i<len; i++)
    {
        if ((i&15) == 0)
        {
            aes_att_encryption (key, r, e);
            r[0]++;
        }
        ps[i] ^= e[i & 15];
    }

    ////////////// calculate mic ///////////////////////
    memset (r, 0, 16);
    memcpy (r, iv, 8);
    r[8] = len;
    aes_att_encryption (key, r, r);

    for (int i=0; i<len; i++)
    {
        r[i & 15] ^= ps[i];

        if ((i&15) == 15 || i == len - 1)
        {
            aes_att_encryption (key, r, r);
        }
    }
```

```
    for (int i=0; i<mic_len; i++)
    {
        if (mic[i] != r[i])
        {
            return 0;                   //Failed
        }
    }
    return 1;
}
```