BLUETOOTH BLOG

Bluetooth Pairing Part 4: Bluetooth Low Energy Secure Connections – Numeric Comparison

January 19, 2017 | Kai Ren | Bluetooth Low Energy

Part 3, we had an overview about LE Legacy pairing with passkey. Other than LE Legacy pairing, LE Secure Connections is the other option for pairing. LE Secure Connections is an enhanced security feature introduced in Bluetooth v4.2. It uses a Federal Information Processing Standards (FIPS) compliant algorithm called Elliptic Curve Diffie Hellman (ECDH) for key generation. For LE Secure Connections, it support four association models:

- **Just Works**
- **Numeric Comparison (Only for LE Secure Connections)**
- **Passkey Entry**
- **Out of Band (OOB)**

Numeric Comparison only exists for LE Secure Connections, not for Legacy pairing. So it's a newer association model, and today we will have an inside look at it.

1. Phase 1 – Pairing Feature Exchange

I had talked about the pairing feature exchange in Part 1, but I want to review it here. Table 1 is the Pairing Request/Response packet definition. In "AuthReq" field, there is one bit named "SC". If LE Secure Connections pairing is supported by the device, then the SC field shall be set to 1, otherwise it shall be set to 0. If both devices support LE Secure Connections pairing, then LE Secure Connections pairing shall be used, otherwise LE Legacy pairing shall be used.

Table 1 Pairing Request/Response

Field	Code (1 Byte)	IO Cap (1 Byte)	OOB Df (1 Byte)	AuthReq (1 Byte)					Maximum Encryption Key Size (1 Byte)	Initiator Key Distribution (1 Byte)	Responder Key Distribution (1 Byte)
				BF	MITM	SC	KP	Reserved			
Bits*	8	8	8	2	1	1	1	3	8	8	8

*Bit order is LSB to MSB.

So if two devices want to pair by LE Secure Connections, they must set "SC" to 1 to indicate peer device that "I have the capability for Secure Connections".

2. Phase 2 – Key Generation Method Selection

After pairing feature exchange, initiator and responder should determine what key generation method will be used. Here is sample C syntax coding for the key generation method:

```
if ( ( 1 == OOB_Flag_Initiator ) || ( 1 == OOB_Flag_Responder) ){
        runOOBMethod();
    }
    else if ( ( 0 == MITM_Flag_Initiator) && ( 0 == MITM_Flag_Responder) ){
        runJustWorksMethod();
    }
    else {
        checkIOCapabilities();
    }
```

Table 2 lists the IO Capabilities of initiating and responding devices for Numeric Comparison. When both the initiating and the responding devices have Display and Yes/No I/O capabilities, or Display and Keyboard I/O capabilities, the Numeric Comparison Association Model is used.

Table 2 Mapping of IO Capabilities for Numeric Comparison

* – mean it is for other key generation methods, other than Numeric Comparison.

3. Phase 2 – Authentication

After key generation, pairing will go to phase 2, Authentication. The aim is for protection against Man-In-The-Middle (MITM) attacks and generation of the keys which will be used to encrypt the connection link.

In public key exchange, each device generates its own Elliptic Curve Diffie-Hellman(ECDH) public-private key pair. The public-private key pair contains a private key and public key.

- *PKa*, private key of initiating device
- *PKa*, public key of initiating device
- *SKb*, private key of responding device
- *PKb*, public key of responding device

Picture 1, the authentication process of Numeric Comparison

Pairing is started by the initiating device sending its PKa to responding device. The responding device replies with its own PKb. After the public keys have been exchanged, the device can then start computing the Diffie-Hellman Key; you can see it starts at the end of 1b in Picture 1.

After that, each device selects a random 128-bit nonce. This value is used to prevent replay attacks.

- *Na*, 128-bit random nonce of initiating device
- *Nb*, 128-bit random nonce of responding device.

Following this the responding device then computes a commitment, *Cb*, which is calculated using *Nb*, *PKa*, *PKb* and 0. It's shown in step 3, Picture 1.

Step 4, the responding device MUST share *Cb* before it receives the initiating device's *Na*.

Step 5, the initiating device MUST share its *Na* before it receives the responding device's *Nb*.

Step 6, the initiating device MUST check *Cb* which is from responding device after it receives the responding device's *Nb*.

At this point, initiating or responding device already knows the peer device's public key and random nonce. The initiating device can confirm commitment (*Cb*) from the responding device. A failure at this point indicates the presence of an attacker or other transmission error and should cause pairing process to abort, step 6.a.

Assuming the commitment check succeeds, the two devices each compute 6-digit confirmation values that are displayed to the user on their respective devices. The user is expected to check that these 6-digit values match and to confirm if there is a match. If no match, pairing aborts.

4. Phase 3 – Long Term Key, LTK

When authentication is successful, the two devices start to compute the LTK which will be used for link encryption. This is the final piece of the puzzle for pairing and reconnection: within different association models, authenticate the peer device and prevent Man in the Middle (MITM) attacks. Since the LTK calculation is common for any LE Secure Connections association model, I will talk about it in greater length in my next blog post.

5. Conclusion

According to the user experience and convenience, compared with Part 3, Passkey Entry, Numeric Comparison just need to two buttons, YES and NO, to indicate whether 6-digit confirmation values match or not between these two devices, it doesn't need a digit-keyboard for passkey input, from '0' to '9', so this is an improvement to simplify the I/O capability for hardware. Meanwhile, due to Numeric Comparison only exist for LE Secure Connections, it provide enhanced protection again the threats like eavesdropping and MITM. So guys, if you start to develop a product which is sensitive of privacy and need highly protection for Bluetooth LE link, here is a good choice for you.

Blog Posts | Member Blogs | Papers | Study Guides | Videos

Blog Post

Best Practices for Using a Standalone Auracast™ Transmitter

The recently released Auracast™ Simple Transmitter Best Practices Guide describes a typical, qualified implementation...

READ MORE

Blog Post

Auracast™ Broadcast Audio Introduces New Opportunities for Product Developers & Public Locations

Bluetooth® technology recently introduced a new Bluetooth capability, Auracast™ broadcast audio, that will deliver life-changing...

READ MORE

Blog Post

Introducing: The Bluetooth Low Energy Primer

Bluetooth® technology has been around for more than 20 years. Initially created to allow...

READ MORE