

BLUETOOTH BLOG

Bluetooth Pairing Part 3 – Low Energy Legacy Pairing Passkey Entry

 August 25, 2016 |  Kai Ren |  Bluetooth Low Energy

In my previous blog on [Key Generation Methods](#), I talked about Key Generation Methods—if the initiating and responding device meet some IO capability conditions, they choose LE legacy Bluetooth pairing Passkey Entry method.

In this blog, I look at legacy pairing with Passkey Entry and how it works.

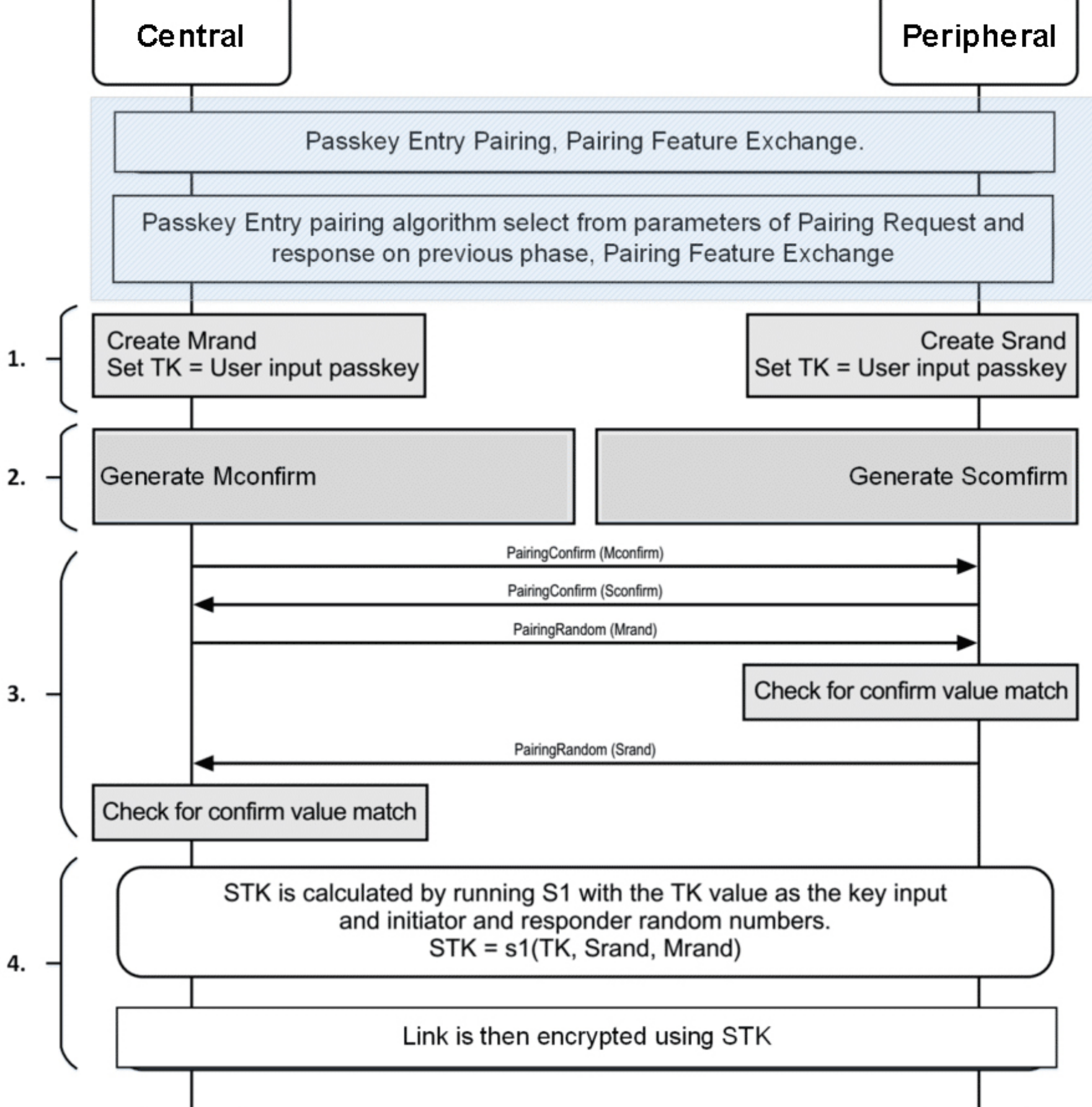


Figure 1: LE Legacy Pairing, Passkey Entry

### Temporary Key (TK) and Random Number Generation

When you use LE legacy pairing, the pairing is performed by each device generating a *Temporary Key (TK)*.

- If the IO capabilities of a device, either the initiating or responding device, has a display capability, then it will display a randomly generated passkey value between “000000” and “999999.” The other device should have an input capability like a keyboard so a user can input the value displayed for the *TK*
- If the IO capabilities of both the initiating and responding devices don't have display capabilities but are both “Keyboard Only,” the user needs to guarantee that the *TKs* between the initiating and responding device are the same. This is a special case for Passkey Entry.

Below is a device named “Authentication” that wants to pair with an iOS device, and it displays *TK* on its output interface. The iOS device then pops up a dialog box and asks the user to input the *TK* value.



Picture 2: Passkey Entry on iOS Device

When the *TK* value is ready, the initiating and responding device generate a 128-bit random number: *Mrand* for the initiating device, *Srand* for the responding device.

### Mconfirm and Sconfirm

*Mconfirm* and *Sconfirm* is 128-bit confirm value which can be calculated using the confirm value generation function *c1*. The detail for this function can be found here: [Bluetooth Core Spec V4.2, Vol.3, Part H, Section 2.2.3](#).

For *c1* function, the input parameters include:

- *TK*
- *Mrand* for *Mconfirm*; or *Srand* for *Sconfirm* calculation
- Pairing Request command
- Pairing Response command
- Initiating device address type
- Initiating device address
- Responding device address type
- Responding device address

### Verification

When *Mconfirm* and *Sconfirm* are ready, the initiating device transmits *Mconfirm* to the responding device. When the responding device receives *Mconfirm* it transmits *Sconfirm* to the initiating device. When the initiating device receives *Sconfirm* it transmits *Mrand* to the responding device.

The responding device verifies the *Mconfirm* value by repeating the calculation the initiating device performed using the *Mrand* value received.

- If the responding device's calculated *Mconfirm* value does not match the received *Mconfirm* value from the initiating device then the pairing process will be aborted and the responding device will send the Pairing Failed command with reason code “Confirm Value Failed.”
- If the responding device's calculated *Mconfirm* value matches the received *Mconfirm* value from the initiating device, the responding device transmits *Srand* to the initiating device.

The initiating device verifies the received *Sconfirm* value by repeating the calculation the responding device performed using the *Srand* value received.

- If the initiating devices calculated *Sconfirm* value does not match the received *Sconfirm* value from the responding device then the pairing process will be aborted and the initiating device will send the Pairing Failed command with the reason code “Confirm Value Failed.”
- If the initiating device's calculated *Sconfirm* value matches the received *Sconfirm* value from the responding device the initiating device then calculates *Short Term Key (STK)* and tells the Controller to enable encryption.

### STK Generation

You generate the STK using the key generation function *s1* detailed in the [Bluetooth Core Spec V4.2, Vol.3, Part H, Section 2.2.4](#).

For *s1* function, the input parameters include:

- *TK*
- *Srand*
- *Mrand*

The paired devices establish an encrypted link with STK.

In [Part 4](#), I introduce a new pairing algorithm in LE Secure Connection: Numeric Comparison.



FEATURED DOWNLOAD

Bluetooth 5: Go Faster, Go Further

Download this comprehensive overview to discover how Bluetooth 5 significantly increases the range, speed, and broadcast messaging capacity of Bluetooth applications, making use cases in smart home automation, enterprise, and industrial markets a reality.

INSTANT DOWNLOAD 


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 

 [About Us](#) [Careers](#) [Contact](#)

[Sign Up for Updates](#) [Join the SIG](#)

© 2022 Bluetooth SIG, Inc. All rights reserved.

[Security](#) | [Privacy Policy](#) | [Terms of Use](#) | [Code of Conduct](#) | [Copyright Policy](#)

[Get Help](#)