Table of contents

BLE Pairing and Bonding

Edited by: Laura Nao

Key concepts

Pairing: process where devices exchange the information necessary to establish an encrypted connection. It involves authenticating the identity of the two devices to be paired, encrypting the link, and distributing keys to allow security to be restarted on a reconnection.

Bonding: process where the information from the pairing process is stored on the devices, so that the pairing process does not have to be repeated every time the devices reconnect to each other.

Security issues

- **Passive eavesdropping**: process by which a third device listens in to the data being exchanged between two paired devices. BLE overcomes this is by encrypting the data being transferred using AES-CCM cryptography + the method by which the keys are exchange (i.e. "pairing method”) has a great effect on the security of the connection.
- **MITM** (Man In The Middle, i.e. active eavesdropping): process by which a third device impersonates the other two legitimate devices, in order to fool them into connecting to it. Both BLE central and peripheral will connect to the malicious device, which in turn routes the communication between the other two devices. The malicious device intercepts all data being sent and can inject false data or remove data before it reaches its recipient. The pairing method used determines how resilient the BLE connection will be to MITM attacks.
- **Identity tracking**: process by which a malicious entity is able to associate the address of a BLE device with a specific user and then physically track that user based upon the presence of the BLE device. The way BLE overcomes this is by periodically changing the device address.

Keys

All security is based on keys (i.e. shared keys).

- **TK** (Temporary Key): used during pairing procedure, set to a value determined by the pairing algorithm. It is used to calculate the STK.
- **STK** (Short Term Key): used as the key for encrypting a connection the very first time two devices pair. The STK is generated using 3 pieces of information:
  - the TK is used as the key for the encryption engine
  - *Srand* random number contributed by the slave
  - *Mrand* random number contributed by the master

$$STK = E_{tk}(Srand | Mrand)$$

- **LTK** (Long Term Key): distributed once the initial pairing procedure has encrypted the connection. It can be:
  - a random number stored in a security DB
  - generated on the slave device
- **IRK** (Identity Resolving Key): gives a device that knows a peer's device IRK the ability to resolve a peer device's identity. A device address is split in 2 parts:
  - random part
  - hash of the random part with the IRK
- **CSRK** (Connection Signature Resolving Key): gives a receiving device the ability to resolve a signature, and therefore authenticate the sender of the message.

Pairing - BLE Legacy VS. LESC

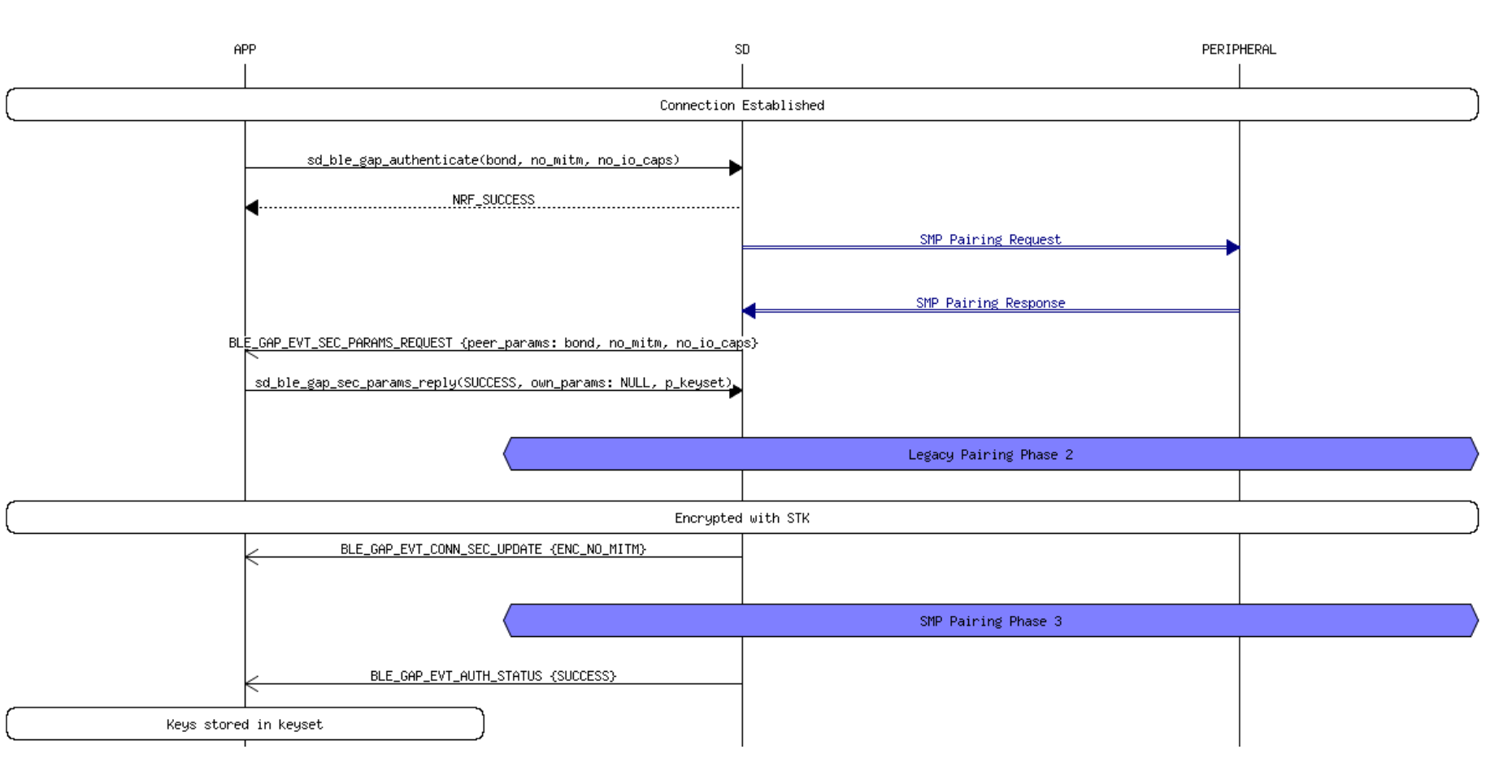
BLE Legacy

The pairing process for 4.0 and 4.1 devices, also known as LE Legacy Pairing, uses a custom key exchange protocol unique to the BLE standard. In this setup, the devices exchange a Temporary Key (TK) and use it to create a Short Term Key (STK) which is used to encrypt the connection. How secure this process is depends greatly on the pairing method used to exchange the TK. The pairing process is performed in a series of phases:

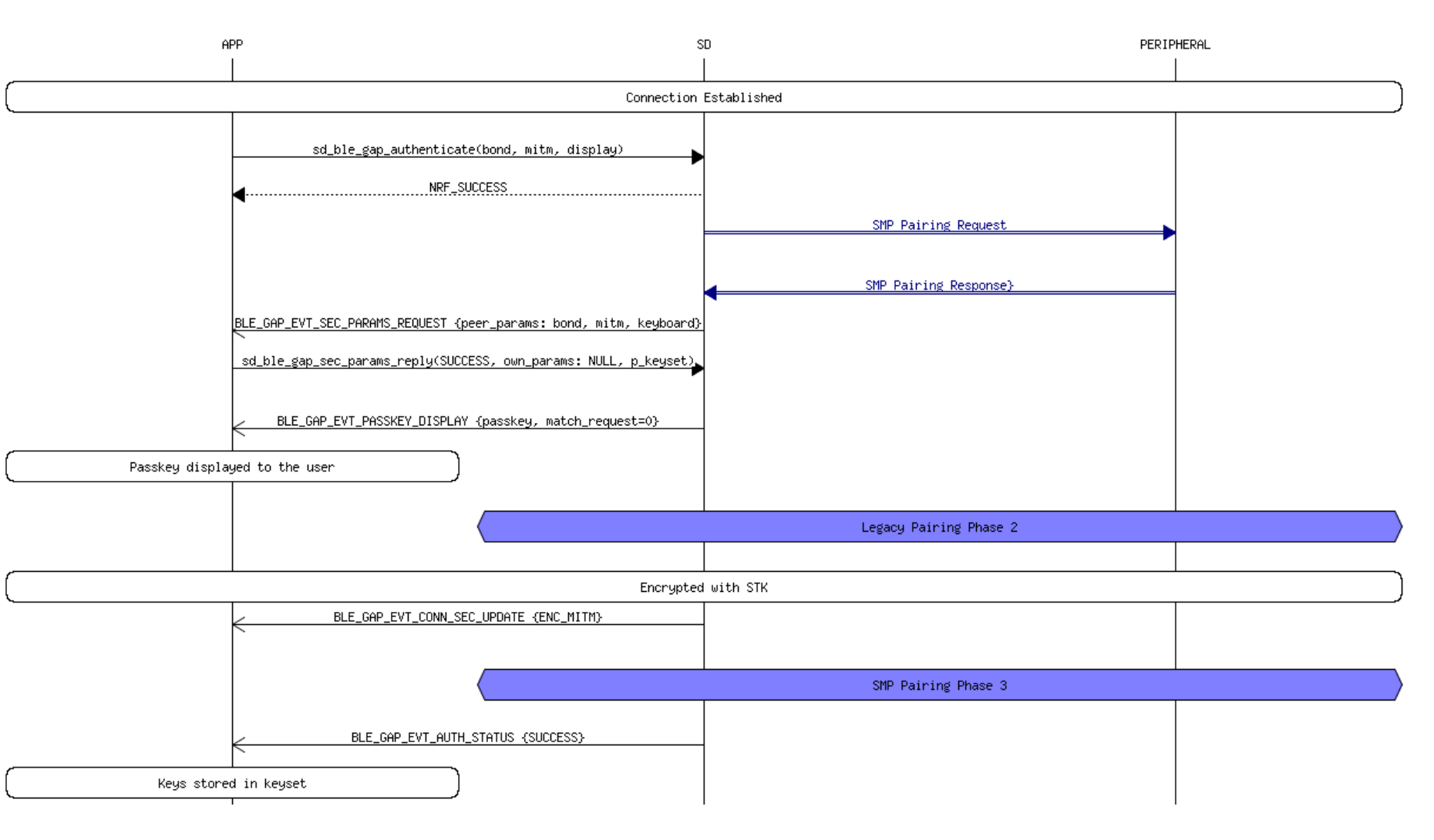
1. This phase begins when the initiating device sends a *Pairing request* to the other device. The two devices then exchange I/O capabilities, authentication requirements, maximum link key size and bonding requirements. Basically all this phase consists of, is the two devices exchanging their capabilities and determining how they are going to go about setting up a secure connection. Note: all data being exchanged during this phase is unencrypted.
2. The devices generate and/or exchange the TK using one of the pairing methods. From there, the two devices then exchange Confirm and Rand values in order to verify that they both are using the same TK. Once this has been determined, they will use the TK along with the Rand values to create the STK. The STK is then used to encrypt the connection.
3. [optional, only used if bonding requirements were exchanged in (1)] Several transport specific keys are exchanged:
   - LTK
   - EDIV, Rand (to create/identify the LTK)
   - CSRK
   - public IEEE address
   - random static address
   - IRK

Pairing methods:

- **Just Works** In this method, the TK is set to 0. Thus, it is very easy for an attacker to brute force the STK and eavesdrop on the connection. Likewise, this method also offers no way of verifying the devices taking part in the connection and thus it offers no MITM protection.



- **Out of Band (OOB)** In this method, the TK is exchanged using a different wireless technology such as NFC. The main advantage to this method is that a very large TK can be used, up to 128 bits, greatly enhancing the security of the connection. If the OOB channel is protected from MITM attacks, then it can be assumed that the BLE connection is also protected from MITM attacks as well. Likewise, as long as the OOB channel is immune to eavesdropping during the pairing process, then the BLE connection will also be immune from passive eavesdropping. Of the three legacy pairing methods, OOB pairing is by far the most secure provided that the OOB channel employs sufficient security methods.
- **Passkey** In this method, the TK is a 6 digit number that is passed between the devices by the user. The way this number is transferred can vary. One example would be to have one of the devices generate a random 6 digit number and display it on a LCD display. The user would then read the number and enter it into the other device using a keypad. If an attacker is not listening in during the pairing process, then the passkey method gives fairly good protection from passive eavesdropping. However, if an attacker is present during the pairing process and is able to sniff the values being exchanged, then it is fairly trivial to brute force the TK and use it to derive the STK and decrypt the connection. The passkey method is generally considered to be secure from MITM attacks provided that the attacker is not able to obtain the passkey via some means other than the BLE connection. However, there is at least one theoretical MITM attack that is able to succeed without advanced knowledge of the passkey.



BLE applications that require the highest level of security should use either the OOB or Numeric Comparison pairing methods.

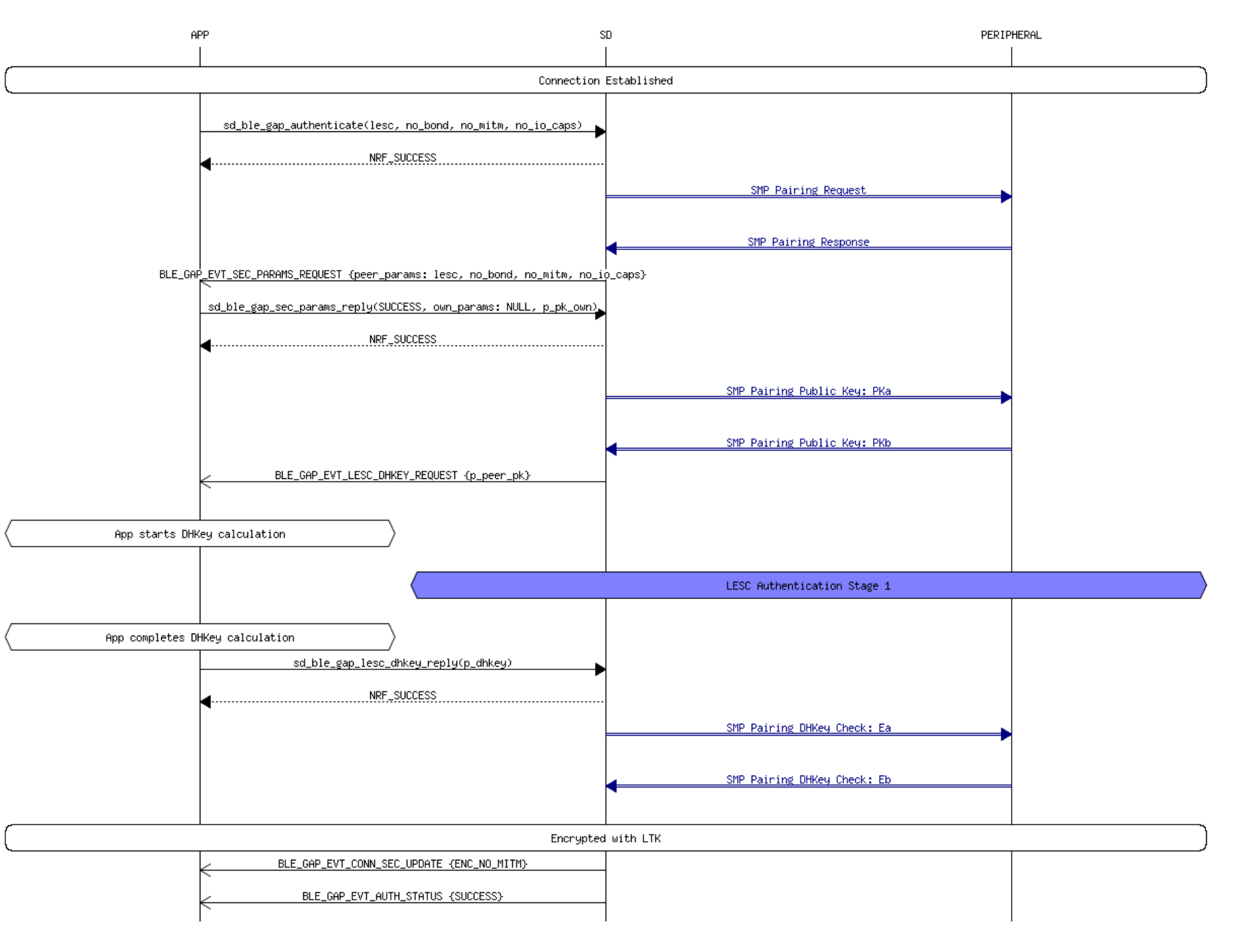
LE Secure Connection

BLE 4.2 devices are fully backwards compatible with BLE 4.0 and 4.1 devices, this means that 4.2 devices are capable performing the exact same pairing process as 4.0 and 4.1 devices. However, BLE 4.2 are also capable of creating what are known as LE Secure Connections. Instead of using a TK and STK, LE Secure Connections use a single Long Term Key (LTK) to encrypt the connection. This LTK is exchanged/generated using **Elliptic Curve Diffie Hellman (ECDH)** public key cryptography which offers significantly stronger security compared to the original BLE key exchange protocol.

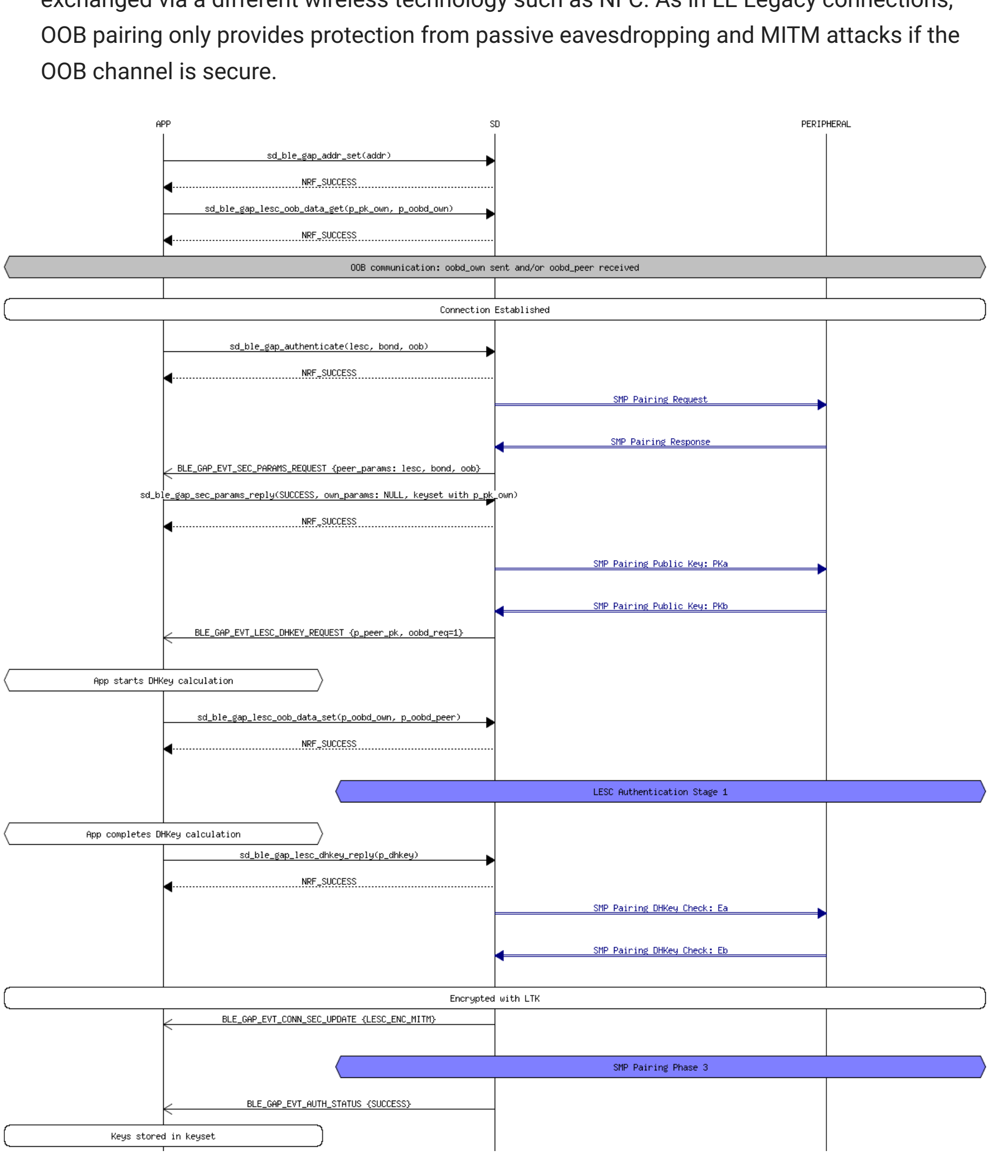
1. Same as in BLE legacy
2. Both devices generate an ECDH public-private key pair. The two devices will then exchange their public keys and then start computing the Diffie-Hellman key. One of the pairing methods is then used to authenticate the connection. Once the connection is authenticated, the LTK is generated and the connection is encrypted.
3. Same as in BLE legacy

Pairing methods:

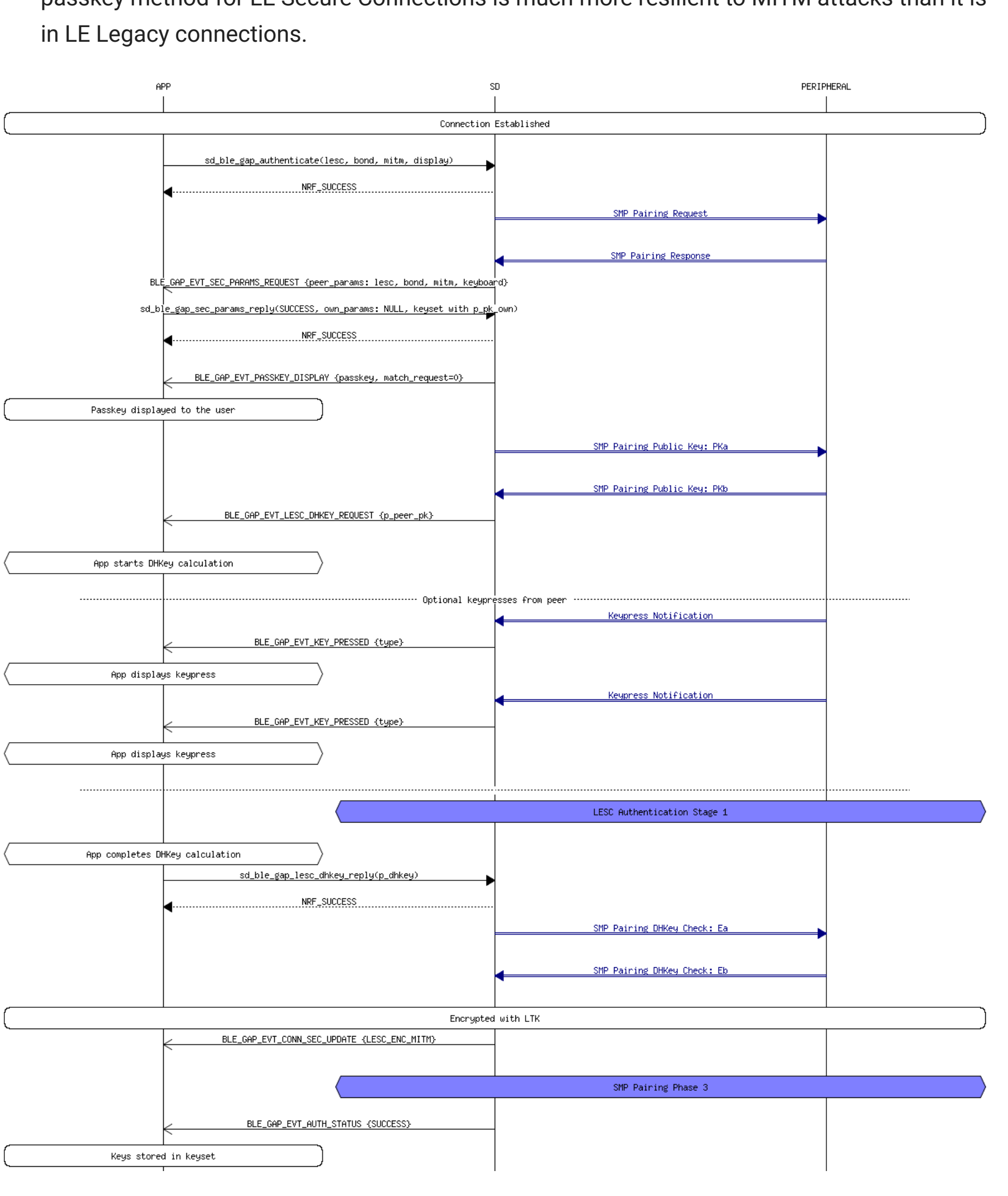
- **Just Works** Once the devices exchange their public keys, the non-initiating device will generate a nonce, which is essentially a random seed value, and then use it to generate a confirmation value Cb. It then sends the Cb along with the nonce to the initiating device. At the same time, the initiating device generates its own nonce and sends it to the non-initiating device. The initiating device then uses the non-initiating device’s nonce to generate its own confirmation value Ca which should match Cb. If the confirmation values match, then the connection proceeds. By virtue of the ECDH key exchange, the Just Works&trade pairing method in LE Secure Connections has substantially more resilience to passive eavesdropping compared to the same method in LE Legacy Connections. However, since this method does not give the user a way to verify the authenticity of the connection, it is still vulnerable to MITM attacks.



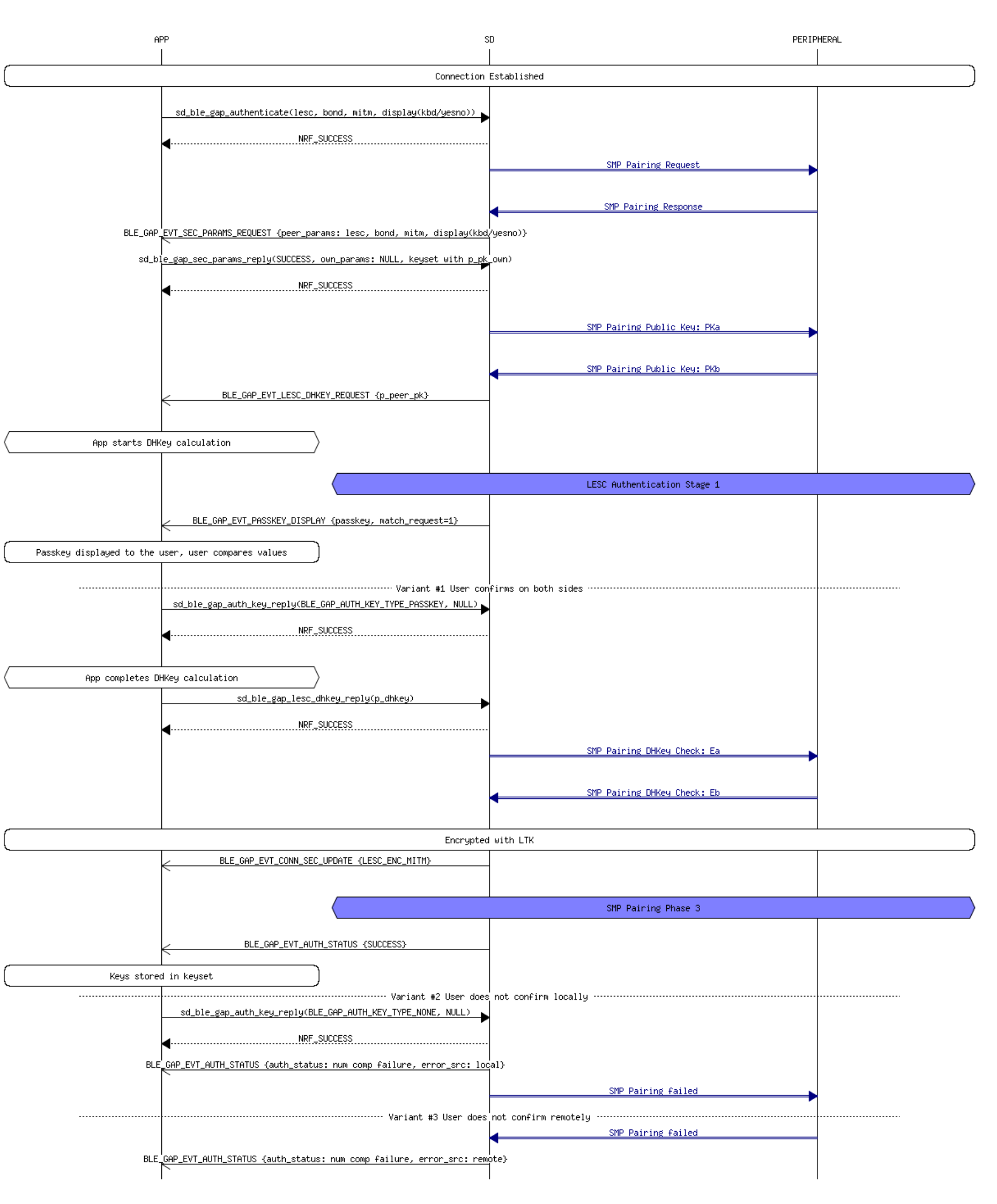
- **Out of Band (OOB)** In OOB pairing, the public keys, nonces and confirmation values are all exchanged via a different wireless technology such as NFC. As in LE Legacy connections, OOB pairing only provides protection from passive eavesdropping and MITM attacks if the OOB channel is secure.



- **Passkey** In this method, an identical 6 digit number is input into each of the devices. The two devices use this passkey, the public keys they exchanged earlier, and a 128 bit nonce to authenticate the connection. This process is done bit by bit for every bit of the passkey. One device will compute a confirmation value for one bit of the passkey and reveal it to the other device. The other device will then compute its own confirmation value for the first bit of its passkey and reveal it to the first device. This process continues until all the bits of the passkey has been exchanged and verified to match. Due to the process detailed above, the passkey method for LE Secure Connections is much more resilient to MITM attacks than it is in LE Legacy connections.



- **Numeric Comparison** This pairing method follows the exact same procedure as the Just WorksTM pairing method, but adds another step at the end. Once the devices confirm that the confirmation values match, then both devices will independently generate a final 6 digit confirmation value using both of the nonces. They both then display their calculated values to the user. The user then manually checks that both values match and ok’s the connection. This extra step allows this pairing method to provide protection from MITM attacks.



GAP connection security modes (Nordic implementation)

```
Security Mode 0 Level 0: No access permissions at all (this level is not defined
Security Mode 1 Level 1: No security is needed (aka open link).
Security Mode 1 Level 2: Encrypted link required, MITM protection not necessary.
Security Mode 1 Level 3: MITM protected encrypted link required.
Security Mode 1 Level 4: LESC MITM protected encrypted link required.
Security Mode 2 Level 1: Signing or encryption required, MITM protection not nec
Security Mode 2 Level 2: MITM protected signing required, unless link is MITM pr
```

Refs:

- *"Bluetooth Low Energy: The Developer's Handbook"*, Robin Heydon
- A Basic Introduction to BLE Security
- Nordic infocenter
- micro-ecc
- ECDH
- Nordic LESC example app