Erratum 10734: Pairing Updates

***Bluetooth*® Erratum**

- **Revision:** v1.0
- **Revision Date:** 2018-07-16
- **Group Prepared By:** Core Specification Working Group (CSWG)
- **Feedback Email:** core-main@bluetooth.org

This Erratum is mandatory and applies to the following specifications (collectively, the “Source Specifications”):

- Core Specification v5.0 [1]
- Core Specification v4.2 [2]
- Core Specification v4.1 [3]
- Core Specification v4.0 [4]
- Core Specification v3.0 + HS [5]
- Core Specification v2.1 + EDR [6]

Abstract:

This document specifies the changes to be applied to the Core Specifications required to incorporate the various pairing updates.

Revision History

Revision Number	Date	Comments
v1.0	2018-07-16	Adopted by the Bluetooth SIG Board of Directors.

Contributors

Name	Company
Joel Linsky	Qualcomm Technologies, Inc.
Marcel Holtmann	Intel Corporation
Alain Michaud	Microsoft Corporation
Mayank Batra	Qualcomm Technologies International, Ltd.
Brian Redding	Qualcomm Technologies, Inc.
Clive D.W. Feather	Samsung Electronics Co., Ltd.
Rasmus Abildgren	Samsung Electronics Co., Ltd.
Shawn Ding	Broadcom Corporation
L.C. Ko	MediaTek Inc.
Pål Håland	Nordic Semiconductor ASA
Martin Turon	Google Inc.
Piotr Winiarczyk	Silvair, Inc.

Contents

1 Language

1.1 Language conventions

Please refer to and follow any terminology, language conventions, and interpretation sections of the Source Specification(s).

2 Conventions used in this Erratum

The formatting and color conventions described in Table 2.1 below are used in this erratum to describe the specific changes and additions to the Source Specification(s) identified on the cover page.

Text Color	Description
black	Text that is unmodified from the Source Specification.
red	Text that is added to the Source Specification.
~~red strikethrough~~	Text that is deleted from the Source Specification.
[green bracketed text]	Comments that are intended to aid the reader.
blue	Default color used for section numbers and headings of this document.

Table 2.1: Color key for headings, captions, and body text

3 Changes to Core Specification v5.0

This Section sets forth the specific changes and additions, using the formatting and color conventions described in Section 2, to Core Specification v5.0.

3.1 Changes to Core Specification v5.0, Volume 2, Part C: Link Manager Protocol Specification

3.1.1 **[Modified Section]** 4.2.7.4 Authentication stage 2: DHKey Check

[The modified section with changes is shown below.]

At this stage, both devices compute new confirmation values based on Diffie-Hellman key and previously exchanged information according to [Vol 2] Part H, Section 7.7.4.

The Initiator shall send an LMP_DHKey_check PDU to the Responder. If the Initiator has determined that the received public key is invalid (see [Vol 2] Part H, Section 7.6), the PDU should include a value that is different from the computed confirmation value (for example, substituting a randomly generated number). Otherwise, the PDU shall include ~~including~~ the computed confirmation value ~~it has computed~~.

[Insert a paragraph break.]

Upon reception, if the received value is not equal to the one calculated according to [Vol 2] Part H, Section 7.7.4, then the Responder shall follow the procedure in Section 4.2.7.4.1. If the values match, the Responder should follow the procedure in Section 4.2.7.4.1 if the received public key is invalid (see [Vol 2] Part H, Section 7.6). Otherwise it shall reply with an LMP_accepted PDU ~~if the received value is equal to the one it has calculated according to Section 7.7.4. If it fails, refer to Section 4.2.7.3.5.1~~.

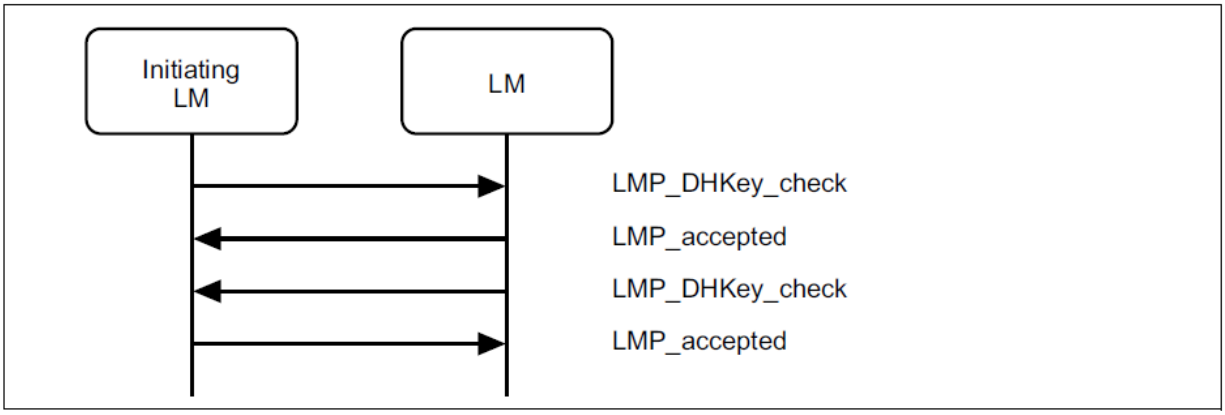
The Responder shall then send an LMP_DHKey_check PDU, ~~to the Initiator~~ including the ~~its~~ confirmation value it has computed, to the Initiator. Upon reception, if the received value is not equal to the one calculated according to [Vol 2] Part H, Section 7.7.4, then the Initiator shall follow the procedure in Section 4.2.7.4.1.1. If the values match, the Initiator should follow the procedure in Section 4.2.7.4.1.1 if the received public key is invalid (see [Vol 2] Part H, Section 7.6). Otherwise it shall reply with an LMP_accepted PDU ~~if the received value is equal to the one it has calculated according to [Vol 2] Part H, Section 7.7.4. If it fails, refer to Section 4.2.7.4.1.1~~.

At this point, both devices shall compute the link key according to [Vol 2] Part H, Section 7.7.3.

If at least one device does not support both the Secure Connections (Controller Support) and the Secure Connections (Host Support) features, the Initiator shall then start standard mutual authentication as described in Section 4.2.1.1

If both devices support both the Secure Connections (Controller Support) and the Secure Connections (Host Support) features, the Initiator shall then start secure authentication as described in Section 4.2.1.4.

After secure authentication, if encryption is enabled, the initiating device shall pause and immediately resume encryption to produce a new encryption key. Note: This will cause a new encryption key to be generated using the h3 function including the ACO created during the secure authentication process.



Sequence 74: DHKey check

A device that detects an invalid public key (see [Vol 2] Part H, Section 7.6) from the peer at any point during the Secure Simple Pairing process shall fail the pairing process and therefore not create a link key.

3.1.2 [Modified Section] *4.2.7.4.1 Check Failure on the Responder Side*

[The modified first paragraph with changes is shown below.]

If the received public key is invalid (see [Vol 2] Part H, Section 7.6), the Responder should send an LMP_not_accepted PDU with reason "Authentication Failure". If the confirmation value received via LMP by the Responder is not equal to the one it has calculated according to [Vol 2] Part H, Section 7.7.4, the Responder shall send an LMP_not_accepted PDU with reason "Authentication Failure".

3.1.3 [Modified Section] *4.2.7.4.1.1 Check Failure on the Initiator Side*

[The modified first paragraph with changes is shown below.]

If the received public key is invalid (see [Vol 2] Part H, Section 7.6), the Initiator should send an LMP_not_accepted PDU with reason "Authentication Failure". If the confirmation value received via LMP by the Initiator is not equal to the one it has calculated according to [Vol 2] Part H, Section 7.7.4, the Initiator shall send an LMP_not_accepted PDU with reason "Authentication Failure".

3.2 Changes to Core Specification v5.0, Volume 2, Part E: Host Controller Interface Functional Specification

3.2.1 [Modified Section] 3 OVERVIEW OF COMMANDS AND EVENTS

[Insert a new row in Table 3.1 as shown below.]

Name	Vers.	Summary description	Supported Controllers

Read Local OOB Extended Data Command	4.1	This command is used to obtain Simple Pairing Hash C and Randomizer R associated with both P-192 and P-256 public keys, which are intended to be transferred to a remote device using an OOB mechanism.	BR/EDR
Read Local Simple Pairing Options	Erratum 10734	The Read Local Simple Pairing Options command is used to read the simple pairing options and the maximum encryption key size supported.	BR/EDR
Read Local Supported Codecs Command	CSA 2	The Read Local Supported Codecs command is used for a Host to query a Controller's supported codecs.	BR/EDR

Table 3.1: Alphabetical list of commands and events

3.2.2 [Modified Section] 6.27 SUPPORTED COMMANDS

[The modified table with changes is shown below.]

Octet	Bit	Command Supported
39	0	LE Read RF Path Compensation Command
	1	LE Write RF Path Compensation Command
	2	LE Set Privacy Mode
	3	Reserved for Future Use
	4	Reserved for Future Use
	5	Reserved for Future Use
	6	Reserved for Future Use
	7	Reserved for Future Use
40	All	Reserved for Future Use
41	0	Reserved for Future Use
	1	Reserved for Future Use
	2	Reserved for Future Use
	3	Read Local Simple Pairing Options
	4	Reserved for Future Use
	5	Reserved for Future Use
	6	Reserved for Future Use
	7	Reserved for Future Use

3.2.3 [New Section] 7.4.9 Read Local Simple Pairing Options Command

[A new section is added as shown below.]

Command	OCF	Command Parameters	Return Parameters
HCI_Read_Local_Simple_Pairing_Options	0x000C		Status, Simple_Pairing_Options, Maximum_Encryption_Key_Size

Description:

The Read_Local_Simple_Pairing_Options command is used to read the simple pairing options and the maximum encryption key size supported. Bit 0 of the Simple_Pairing_Options return parameter shall be set to 1.

Note: If this command is supported, then the Controller must support remote public key validation (see [Vol 2] Part H, Section 7.6).

Command parameters:

None

Return parameters:

Status: *Size: 1 octet*

Value	Parameter Description
0x00	Read_Local_Simple_Pairing_Options command succeeded.
0x01–0xFF	Read_Local_Simple_Pairing_Options command failed. See [Vol 2] Part D, Error Codes for a list of error codes and descriptions.

Simple_Pairing_Options: *Size: 1 octet*

Bit Number	Parameter Description
0	Remote public key validation is always performed.
All other bits	Reserved for future use.

Maximum_Encryption_Key_Size: *Size: 1 octet*

Value	Parameter Description
0x07–0x10	Maximum encryption key size (in octets) supported.
All other values	Reserved for future use.

Event(s) generated (unless masked away):

When the Read_Local_Simple_Pairing_Options command has completed, a Command Complete event shall be generated.

3.2.4 **[Modified Section]** *7.7.65.9 LE Generate DHKey Complete Event*

[The modified Description with changes is shown below.]

Description:

This event indicates that LE Diffie Hellman key generation has been completed by the Controller.

If the Remote_P-256_Public_Key parameter of the HCI_LE_Generate_DHKey command (see Section 7.8.37) was invalid (see [Vol 3] Part H, Section 2.3.5.6.1), then all octets of the DHKey event parameter should be set to 0xFF.

3.2.5 [Modified Section] 7.8.37 LE Generate DHKey Command

[The modified Description with changes is shown below.]

Description:

The LE_Generate_DHKey command is used to initiate generation of a Diffie-Hellman key in the Controller for use over the LE transport. This command takes the remote P-256 public key as input. The Diffie-Hellman key generation uses the private key generated by LE_Read_Local_P256_Public_Key command.

The Diffie-Hellman key returned via this command shall not be generated using any keys used for Secure Connections over the BR/EDR transport.

If the remote P-256 public key is invalid (see [Vol 3] Part H, Section 2.3.5.6.1), the Controller shall return an error and should use the error code *Invalid HCI Command Parameters* (0x12).

3.3 Changes to Core Specification v5.0, Volume 2, Part H: Security Specification

3.3.1 [Modified Section] 5.1 REPEATED ATTEMPTS

[The modified section with changes is shown below.]

When the authentication attempt fails, a waiting interval shall pass before the verifier will initiate a new authentication attempt to the same claimant, or before it will respond to an authentication attempt initiated by a device claiming the same identity as the failed device. For each subsequent authentication failure, the waiting interval shall be increased exponentially. For example, after each failure, the waiting interval before a new attempt can be made could be twice as long as the waiting interval prior to the previous attempt[1]. The waiting interval shall be limited to a maximum.

The maximum waiting interval depends on the implementation. The waiting time shall exponentially decrease to a minimum when no new failed attempts are made during a certain time period. This procedure prevents an intruder from repeating the authentication procedure with a large number of different keys.

To protect a device's private key, a device should implement a method to prevent an attacker from retrieving useful information about the device's private key ~~using invalid public keys~~. For this purpose, a device should change its private key after every pairing (successful or failed). Otherwise, it should change its private key at least after any of the following ~~can use one of the following methods~~:

- ~~Change its private key after~~ three failed attempts from any BD_ADDR ~~and~~
- ~~after 10~~ ten successful pairings from any BD_ADDR ~~; or~~

• ~~after~~ any combination of these such that ~~3~~ three successful pairings count as one failed pairing~~.; or~~

~~• Verify that the received public keys from any BD_ADDR are on the correct curve; or~~

~~• Implement elliptic curve point addition and doubling using formulas that are valid only on the correct curve.~~

3.3.2 [Modified Section] 7.1 PHASE 1: PUBLIC KEY EXCHANGE

[The modified section with changes is shown below.]

Initially, each device generates its own Elliptic Curve Diffie-Hellman (ECDH) public-private key pair (step 1). ~~This key pair needs to be generated only once per device and may be computed in advance of pairing. A device may, at any time, choose to discard its public-private key pair and generate a new one, although there is not a requirement to do so.~~ See Section 5.1 for recommendations on how frequently this key pair should be changed.

Pairing is initiated by the initiating device sending its public key to the receiving device (step 1a). The responding device replies with its own public key (step 1b) These public keys are not regarded as secret although they may identify the devices. Note that steps 1a and 1b are the same in all three protocols.

When both device's Controllers and Hosts support Secure Connections, the P-256 elliptic curve is used. When at least one device's Controller or Host doesn't support Secure Connections, the P-192 elliptic curve is used.

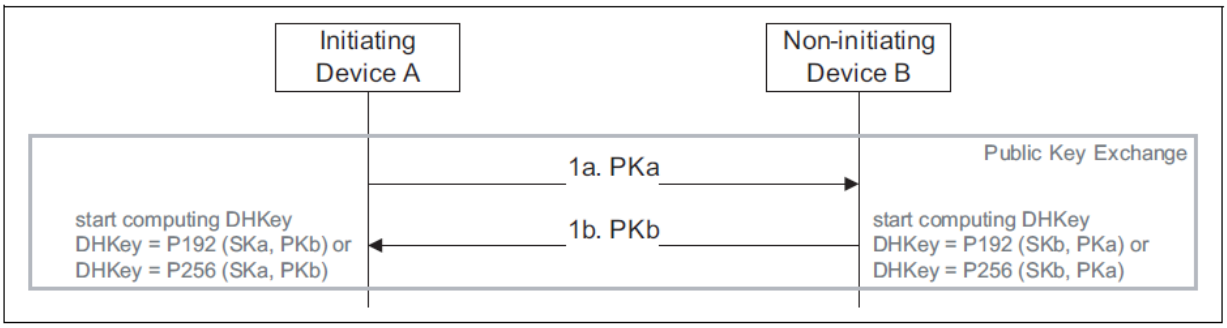


Figure 7.2: Public Key Exchange Details

A device shall validate that any public key received from any BD_ADDR is on the correct curve (P-192 or P-256) – see Section 7.6.

3.3.3 [Modified Section] 7.6 ELLIPTIC CURVE DEFINITION

[The modified text with changes is shown below. Two new paragraphs have been inserted at the end.]

For P-256:

```
p = 11579208921035624876269744694940757353008614341529031419553363130
8867097853951
r = 11579208921035624876269744694940757352999695522413576034242225906
1068512044369
```

```
b = 5ac635d8 aa3a93e7 b3ebbd55 769886bc 651d06b0 cc53b0f6 3bce3c3e
27d2604b
Gx = 6b17d1f2 e12c4247 f8bce6e5 63a440f2 77037d81 2deb33a0 f4a13945
d898c296
Gy = 4fe342e2 fe1a7f9b 8ee7eb4a 7c0f9e16 2bce3357 6b315ece cbb6406837bf51f5
```

The function P-256 is defined as follows. Given an integer u, 0 < u < r, and a point V on the curve E, the value P-256(u,V) is computed as the x-coordinate of the u^{th} multiple uV of the point V.

The private keys shall be between 1 and r/2, where r is the Order of the Abelian Group on the elliptic curve (e.g. between 1 and $2^{256}/2$).

A valid public key Q = (X_Q, Y_Q) is one where X_Q and Y_Q are both in the range 0 to p - 1 and satisfy the equation $(Y_Q)^2 = (X_Q)^3 + aX_Q + b \pmod p$ in the relevant curve's finite field.

A device can validate a public key by directly checking the curve equation, by implementing elliptic curve point addition and doubling using formulas that are valid only on the correct curve, or by other means.

3.4 Changes to Core Specification v5.0, Volume 3, Part C: Generic Access Profile

3.4.1 **[Modified Section]** 14.1 CROSS-TRANSPORT KEY DERIVATION

[The modified section with changes is shown below.]

If both the local and remote devices support Secure Connections over the BR/EDR and LE transports, devices may optionally generate keys of identical strength and the same MITM protection for both transports as part of a single pairing procedure.

If both the local and remote devices support Secure Connections over the LE transport but not over the BR/EDR transport, then the devices may optionally generate the BR/EDR keys of identical strength and the same MITM protection as the LE keys as part of the LE pairing procedure.

If Secure Connections pairing occurs first on the LE transport the procedures in [Vol 3] Part H, Section 2.3.5.7 may be used.

If Secure Connections pairing occurs first on the BR/EDR transport the procedures in [Vol 3] Part H, Section 2.3.5.7 may be used.

If the BR/EDR link key has been generated by a Controller that does not perform remote public key validation (see [Vol 2] Part H, Section 7.6), then the LE LTK should not be generated from such a BR/EDR link key using cross-transport key derivation.

Note: The Host can use the HCI_Read_Local_Simple_Pairing_Options command (see [Vol 2] Part E, Section 7.4.X) or vendor-specific methods to determine whether the Controller performs remote public key validation.

If the LE LTK has been generated using the HCI_LE_Generate_DHKey command (see [Vol 2] Part E, Section 7.8.37) by a Controller that does not perform remote public key validation (see [Vol 3] Part H, Section 2.3.5.6.1), then the BR/EDR link key should not be generated from such an LE LTK using cross-transport key derivation.

Note: The Host can use the Remote Public Key Validation feature bit (see [Vol 6] Part B, Section 4.6) or vendor-specific methods to determine whether the HCI_LE_Generate_DHKey command performs the remote public key validation.

3.5 Changes to Core Specification v5.0, Volume 3, Part H: Security Manager Specification

3.5.1 [Modified Section] *2.3.5.6.1 Public Key Exchange*

[The modified section with changes is shown below.]

Initially, each device generates its own Elliptic Curve Diffie-Hellman (ECDH) public-private key pair (phase 1). The public-private key pair contains a private (secret) key, and a public key. The private keys of devices A and B are denoted as SKa and SKb respectively. The public keys of devices A and B ~~and~~ are denoted as PKa and PKb respectively. ~~This key pair needs to be generated only once per device and may be computed in advance of pairing. A device may, at any time, choose to discard its public-private key pair and generate a new one, although there is not a requirement to do so.~~ See Section 2.3.6 for recommendations on how frequently this key pair should be changed.

Pairing is initiated by the initiating device sending its public key to the receiving device (phase 1a). The responding device replies with its own public key (phase 1b) These public keys are not regarded as secret although they may identify the devices. Note that phases 1a and 1b are the same in all three protocols.

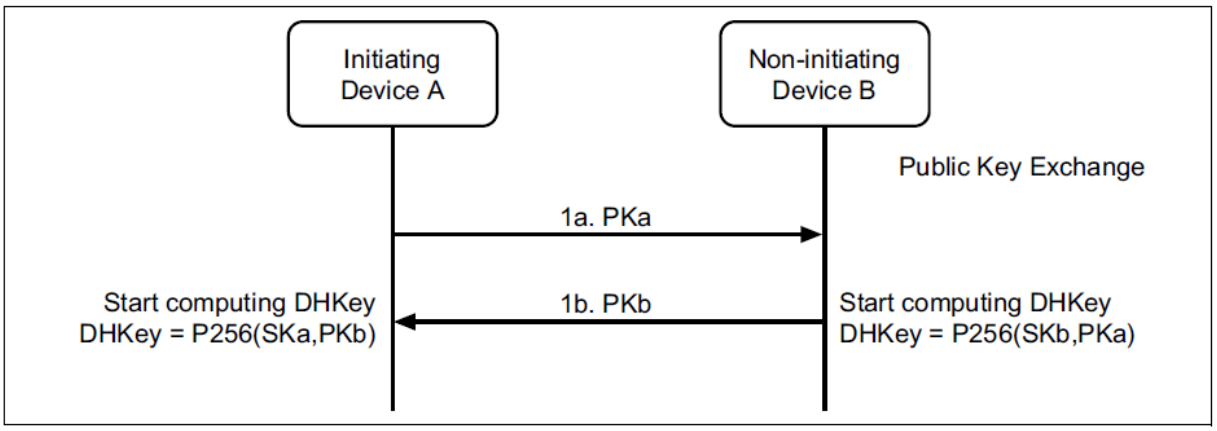


Figure 2.2: Public Key Exchange

A device shall validate that any public key received from any BD_ADDR is on the correct curve (P-256).

A valid public key $Q = (X_Q, Y_Q)$ is one where X_Q and Y_Q are both in the range 0 to p - 1 and satisfy the equation $(Y_Q)^2 = (X_Q)^3 + aX_Q + b \pmod p$ in the relevant curve's finite field. See [Vol 2] Part H, Section 7.6 for the values of a, b, and p.

A device can validate a public key by directly checking the curve equation, by implementing elliptic curve point addition and doubling using formulas that are valid only on the correct curve, or by other means.

A device that detects an invalid public key from the peer at any point during the LE Secure Connections pairing process shall not use the resulting LTK, if any.

After the public keys have been exchanged, the device can then start computing the Diffie-Hellman Key.

When the Security Manager is placed in a Debug mode it shall use the following Diffie-Hellman private / public key pair:

Private key: `3f49f6d4 a3c55f38 74c9b3e3 d2103f50 4aff607b eb40b799 5899b8a6 cd3c1abd`
Public key (X): `20b003d2 f297be2c 5e2c83a7 e9f9a5b9 eff49111 acf4fddb cc030148 0e359de6`
Public key (Y): `dc809c49 652aeb6d 63329abf 5a52155c 766345c2 8fed3024 741c8ed0 1589d28b`

Note: Only one side (initiator or responder) needs to set Secure Connections debug mode in order for debug equipment to be able to determine the LTK and, therefore, be able to monitor the encrypted connection.

3.5.2 [Modified Section] 2.3.6 Repeated Attempts

[The modified section with changes is shown below.]

When a pairing procedure fails a waiting interval shall pass before the verifier will initiate a new Pairing Request command or Security Request command to the same claimant, or before it will respond to a Pairing Request command or Security Request command initiated by a device claiming the same identity as the failed device. For each subsequent failure, the waiting interval shall be increased exponentially. That is, after each failure, the waiting interval before a new attempt can be made, could be for example, twice as long as the waiting interval prior to the previous attempt[1]. The waiting interval should be limited to a maximum.

The maximum waiting interval depends on the implementation. The waiting time shall exponentially decrease to a minimum when no new failed attempts are made during a certain time period. This procedure prevents an intruder from repeating the pairing procedure with a large number of different keys.

To protect a device's private key, a device should implement a method to prevent an attacker from retrieving useful information about the device's private key ~~using invalid public keys~~. For this purpose, a device should change its private key after every pairing (successful or failed). Otherwise, it should change its private key at least after any of the following ~~can use one of the following methods~~:

- ~~Change its private key after~~ three failed attempts from any BD_ADDR ~~and~~
- ~~after 10~~ ten successful pairings from any BD_ADDR ~~; or~~
- ~~after~~ any combination of these such that ~~3~~ three successful pairings count as one failed pairing ~~.; or~~
- ~~Verify that the received public keys from any BD_ADDR are on the correct curve; or~~
- ~~Implement elliptic curve point addition and doubling using formulas that are valid only on the correct curve.~~

[1] Another appropriate integer value larger than 1 may be used.

3.5.3 [Modified Section] 3.5.5 Pairing Failed

[The modified text with changes is shown below. A new paragraph has been inserted after the third paragraph and before Figure 3.7.]

This is used when there has been a failure during pairing and reports that the pairing procedure has been stopped and no further communication for the current pairing procedure is to occur. The Pairing Failed command is defined in Figure 3.7.

Any subsequent pairing procedure shall restart from the Pairing Feature Exchange phase.

This command may be sent at any time during the pairing process by either device in response to a message from the remote device.

During LE Secure Connections pairing, this command should be sent if the remote device's public key is invalid (see [Vol 3] Part H, Section 2.3.5.6.1). The Reason field should be set to "DHKey Check Failed".

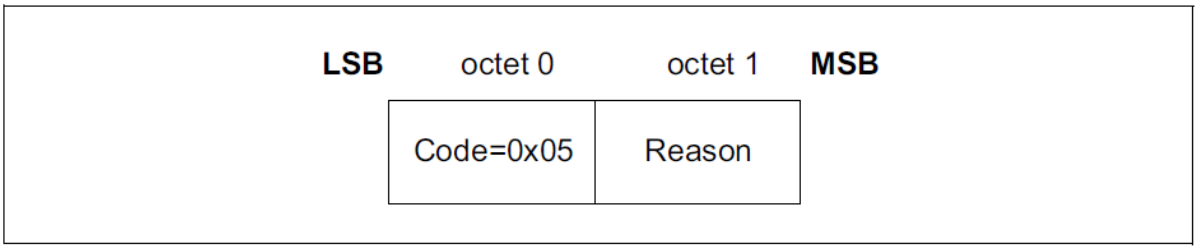


Figure 3.7: Pairing Failed Packet

3.6 Changes to Core Specification v5.0, Volume 6, Part B: Link Layer Specification

3.6.1 [Modified Section] 4.6 FEATURE SUPPORT

[The modified section with changes is shown below.]

The set of features supported by a Link Layer is represented by a bit mask called FeatureSet. The value of FeatureSet shall not change while the Controller has a connection to another device. A peer device may cache information about features that the device supports. The Link Layer may cache information about features that a peer supports during a connection.

Within FeatureSet, a bit set to 0 indicates that the Link Layer Feature is not supported in this Controller; a bit set to 1 indicates that the Link Layer Feature is supported in this Controller.

A Link Layer shall not use a procedure that is not supported by the peer's Link Layer. A Link Layer shall not transmit a PDU listed in the following subsections unless it supports at least one of the features that requires support for that PDU.

The bit positions for each Link Layer Feature shall be as shown in Table 4.4. This table also shows if these bits are valid between Controllers and which bits are masked before transmission to the peer

device. If a bit is shown as not valid, using 'N', then this bit shall be ignored upon receipt by the peer Controller. If a bit is shown as masked, using 'Y', then this bit shall be set to zero when FeatureSet is sent to the peer device; otherwise, it shall be left unchanged.

Bit position	Link Layer Feature	Valid from Controller to Controller	Masked to Peer
0	LE Encryption	Y	N
1	Connection Parameters Request Procedure	Y	N
2	Extended Reject Indication	Y	N
3	Slave-initiated Features Exchange	Y	N
4	LE Ping	N	N
5	LE Data Packet Length Extension	Y	N
6	LL Privacy	N	N
7	Extended Scanner Filter Policies	N	N
8	LE 2M PHY	Y	N
9	Stable Modulation Index - Transmitter	Y	N
10	Stable Modulation Index - Receiver	Y	N
11	LE Coded PHY	Y	N
12	LE Extended Advertising	N	N
13	LE Periodic Advertising	N	N
14	Channel Selection Algorithm #2	Y	N
15	LE Power Class 1	Y	N
16	Minimum Number of Used Channels Procedure		N
27	Remote Public Key Validation	N	Y
All other ~~values~~bits	Reserved for Future Use		

Table 4.4: FeatureSet field's bit mapping to Controller features

3.6.2 [New Section] 4.6.23 Remote Public Key Validation

[A new section is added as shown below.]

A Controller that supports Remote Public Key Validation shall validate the remote public key (see [Vol 3] Part H, Section 2.3.5.6.1) sent by the Host in the HCI_LE_Generate_DHKey command (see [Vol 2] Part E, Section 7.8.37).

3.6.3 [Modified Section] 5.1.4 Feature Exchange Procedure

[The modified section with changes is shown below.]

The Link Layer parameter for the current supported feature set (FeatureSet) may be exchanged after entering the Connection State. Both the master and slave can initiate this procedure.

The FeatureSet information may be cached either during a connection or between connections. A Link Layer should not request this information on every connection if the information has been cached for this device. Cached information for a device from a previous connection is not authoritative and, therefore, an implementation must be able to accept the LL_UNKNOWN_RSP PDU if use of a feature is attempted that is not currently supported or used by the peer.

The $FeatureSet_M$ parameter is the feature capabilities of the Link Layer of the master with certain bits masked as specified in Section 4.6.

The $FeatureSet_S$ parameter is the feature capabilities of the Link Layer of the Slave with certain bits masked as specified in Section 4.6.

The $FeatureSet_{USED}$ parameter is one octet long and is the logical AND of $FeatureSet_M[0]$ and $FeatureSet_S[0]$.

4 Changes to Core Specification v4.2

This Section sets forth the specific changes and additions, using the formatting and color conventions described in Section 2, to Core Specification v4.2.

4.1 Changes to Core Specification v4.2, Volume 2, Part C: Link Manager Protocol Specification

4.1.1 **[Modified Section]** *4.2.7.4 Authentication stage 2: DHKey Check*

[The modified section with changes is shown below.]

At this stage, both devices compute new confirmation values based on Diffie-Hellman key and previously exchanged information according to [Vol 2] Part H, Section 7.7.4, “The Simple Pairing Check Function f3,” on page 1354.

The Initiator shall send an LMP_DHKey_check PDU to the Responder. If the Initiator has determined that the received public key is invalid (see [Vol 2] Part H, Section 7.6), the PDU should include a value that is different from the computed confirmation value (for example, substituting a randomly generated number). Otherwise, the PDU shall include ~~including~~ the computed confirmation value ~~it has computed~~.

[Insert a paragraph break.]

Upon reception, if the received value is not equal to the one calculated according to [Vol 2] Part H, Section 7.7.4, then the Responder shall follow the procedure in Section 4.2.7.4.1. If the values match, the Responder should follow the procedure in Section 4.2.7.4.1 if the received public key is invalid (see [Vol 2] Part H, Section 7.6). Otherwise it shall reply with an LMP_accepted PDU ~~if the received value is equal to the one it has calculated according to Section 7.7.4 on page 1354. If it fails, refer to Section 4.2.7.3.5.1 on page 305~~.

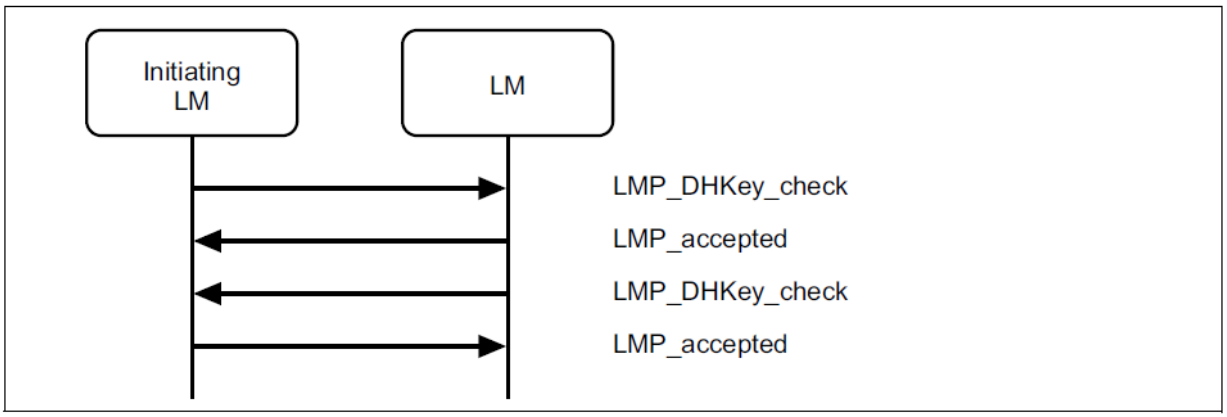
The Responder shall then send an LMP_DHKey_check PDU, ~~to the Initiator~~ including the ~~its~~ confirmation value it has computed, to the Initiator. Upon reception, if the received value is not equal to the one calculated according to [Vol 2] Part H, Section 7.7.4, then the Initiator shall follow the procedure in Section 4.2.7.4.1.1. If the values match, the Initiator should follow the procedure in Section 4.2.7.4.1.1 if the received public key is invalid (see [Vol 2] Part H, Section 7.6). Otherwise it shall reply with an LMP_accepted PDU ~~if the received value is equal to the one it has calculated according to Section 7.7.4 on page 1354. If it fails, refer to Section 4.2.7.4.1.1, “Check Failure on the Initiator Side,” on page 308~~.

At this point, both devices shall compute the link key according to [Vol 2] Part H, Section 7.7.3, “The Simple Pairing Key Derivation Function f2,” on page 1353.

If at least one device does not support both the Secure Connections (Controller Support) and the Secure Connections (Host Support) features, the Initiator shall then start standard mutual authentication as described in Section 4.2.1.1

If both devices support both the Secure Connections (Controller Support) and the Secure Connections (Host Support) features, the Initiator shall then start secure authentication as described in Section 4.2.1.4.

After secure authentication, if encryption is enabled, the initiating device shall pause and immediately resume encryption to produce a new encryption key. Note: This will cause a new encryption key to be generated using the h3 function including the ACO created during the secure authentication process.



Sequence 71: DHKey check

A device that detects an invalid public key (see [Vol 2] Part H, Section 7.6) from the peer at any point during the Secure Simple Pairing process shall fail the pairing process and therefore not create a link key.

4.1.2 [Modified Section] *4.2.7.4.1 Check Failure on the Responder Side*

[The modified first paragraph with changes is shown below.]

If the received public key is invalid (see [Vol 2] Part H, Section 7.6), the Responder should send an LMP_not_accepted PDU with reason “Authentication Failure”. If the confirmation value received via LMP by the Responder is not equal to the one it has calculated according to [Vol 2] Part H, Section 7.7.4 on page 1354, the Responder shall send an LMP_not_accepted PDU with reason “Authentication Failure”.

4.1.3 [Modified Section] *4.2.7.4.1.1 Check Failure on the Initiator Side*

[The modified first paragraph with changes is shown below.]

If the received public key is invalid (see [Vol 2] Part H, Section 7.6), the Initiator should send an LMP_not_accepted PDU with reason “Authentication Failure”. If the confirmation value received via LMP by the Initiator is not equal to the one it has calculated according to [Vol 2] Part H, Section 7.7.4 on page 1354, the Initiator shall send an LMP_not_accepted PDU with reason “Authentication Failure”.

4.2 Changes to Core Specification v4.2, Volume 2, Part E: Host Controller Interface Functional Specification

4.2.1 [Modified Section] 3.4 CONTROLLER INFORMATION

[Insert a new row at the end of Table 3.5 as shown below.]

Name	Vers.	Summary description	Supported Controllers

LE Read Maximum Data Length Command	4.2	The LE Read Maximum Data Length command allows the Host to read the Controller's supportedMaxTxOctets, supportedMaxTxTime, supportedMaxRxOctets, and supportedMaxRxTime parameters.	LE
Read Local Simple Pairing Options	Erratum 10734	The Read Local Simple Pairing Options command is used to read the simple pairing options and the maximum encryption key size supported.	BR/EDR

Table 3.5: Controller information

4.2.2 [Modified Section] 6.27 SUPPORTED COMMANDS

[The modified table with changes is shown below.]

Octet	Bit	Command Supported
35	0	LE Read Local Resolvable Address
	1	LE Set Address Resolution Enable
	2	LE Set Resolvable Private Address Timeout
	3	LE Read Maximum Data Length
	4	Reserved
	5	Reserved
	6	Reserved
	7	Reserved
36	All	Reserved
37	All	Reserved
38	All	Reserved
39	All	Reserved
40	All	Reserved
41	0	Reserved
	1	Reserved
	2	Reserved
	3	Read Local Simple Pairing Options
	4	Reserved
	5	Reserved
	6	Reserved
	7	Reserved

4.2.3 [New Section] 7.4.9 Read Local Simple Pairing Options Command

[A new section is added as shown below.]

Command	OCF	Command Parameters	Return Parameters
HCI_Read_Local_Simple_Pairing_Options	0x000C		Status, Simple_Pairing_Options, Maximum_Encryption_Key_Size

Description:

The Read_Local_Simple_Pairing_Options command is used to read the simple pairing options and the maximum encryption key size supported. Bit 0 of the Simple_Pairing_Options return parameter shall be set to 1.

Note: If this command is supported, then the Controller must support remote public key validation (see [Vol 2] Part H, Section 7.6).

Command parameters:

None

Return parameters:

Status: *Size: 1 octet*

Value	Parameter Description
0x00	Read_Local_Simple_Pairing_Options command succeeded.
0x01–0xFF	Read_Local_Simple_Pairing_Options command failed. See [Vol 2] Part D, Error Codes for a list of error codes and descriptions.

Simple_Pairing_Options: *Size: 1 octet*

Bit Number	Parameter Description
0	Remote public key validation is always performed.
All other bits	Reserved for future use.

Maximum_Encryption_Key_Size: *Size: 1 octet*

Value	Parameter Description
0x07–0x10	Maximum encryption key size (in octets) supported.
All other values	Reserved for future use.

Event(s) generated (unless masked away):

When the Read_Local_Simple_Pairing_Options command has completed, a Command Complete event shall be generated.

4.2.4 **[Modified Section]** *7.7.65.9 LE Generate DHKey Complete Event*

[The modified Description with changes is shown below.]

Description:

This event indicates that LE Diffie Hellman key generation has been completed by the Controller.

If the Remote_P-256_Public_Key parameter of the HCI_LE_Generate_DHKey command (see Section 7.8.37) was invalid (see [Vol 3] Part H, Section 2.3.5.6.1), then all octets of the DHKey event parameter should be set to 0xFF.

4.2.5 [Modified Section] 7.8.37 LE Generate DHKey Command

[The modified Description with changes is shown below.]

Description:

The LE_Generate_DHKey command is used to initiate generation of a Diffie-Hellman key in the Controller for use over the LE transport. This command takes the remote P-256 public key as input. The Diffie-Hellman key generation uses the private key generated by LE_Read_Local_P256_Public_Key command.

The Diffie-Hellman key returned via this command shall not be generated using any keys used for Secure Connections over the BR/EDR transport.

If the remote P-256 public key is invalid (see [Vol 3] Part H, Section 2.3.5.6.1), the Controller shall return an error and should use the error code *Invalid HCI Command Parameters* (0x12).

4.3 Changes to Core Specification v4.2, Volume 2, Part H: Security Specification

4.3.1 [Modified Section] 5.1 REPEATED ATTEMPTS

[The modified section with changes is shown below.]

When the authentication attempt fails, a waiting interval shall pass before the verifier will initiate a new authentication attempt to the same claimant, or before it will respond to an authentication attempt initiated by a device claiming the same identity as the failed device. For each subsequent authentication failure, the waiting interval shall be increased exponentially. For example, after each failure, the waiting interval before a new attempt can be made could be twice as long as the waiting interval prior to the previous attempt[1]. The waiting interval shall be limited to a maximum.

The maximum waiting interval depends on the implementation. The waiting time shall exponentially decrease to a minimum when no new failed attempts are made during a certain time period. This procedure prevents an intruder from repeating the authentication procedure with a large number of different keys.

To protect a device's private key, a device should implement a method to prevent an attacker from retrieving useful information about the device's private key ~~using invalid public keys~~. For this purpose, a device should change its private key after every pairing (successful or failed). Otherwise, it should change its private key at least after any of the following ~~can use one of the following methods~~:

- ~~Change its private key after~~ three failed attempts from any BD_ADDR ~~and~~
- ~~after 10~~ ten successful pairings from any BD_ADDR ~~; or~~

• ~~after~~ any combination of these such that ~~3~~ three successful pairings count as one failed pairing~~.; or~~

~~• Verify that the received public keys from any BD_ADDR are on the correct curve; or~~

~~• Implement elliptic curve point addition and doubling using formulas that are valid only on the correct curve.~~

4.3.2 [Modified Section] 7.1 PHASE 1: PUBLIC KEY EXCHANGE

[The modified section with changes is shown below.]

Initially, each device generates its own Elliptic Curve Diffie-Hellman (ECDH) public-private key pair (step 1). ~~This key pair needs to be generated only once per device and may be computed in advance of pairing. A device may, at any time, choose to discard its public-private key pair and generate a new one, although there is not a requirement to do so.~~ See Section 5.1 for recommendations on how frequently this key pair should be changed.

Pairing is initiated by the initiating device sending its public key to the receiving device (step 1a). The responding device replies with its own public key (step 1b) These public keys are not regarded as secret although they may identify the devices. Note that steps 1a and 1b are the same in all three protocols.

When both device's Controllers and Hosts support Secure Connections, the P-256 elliptic curve is used. When at least one device's Controller or Host doesn't support Secure Connections, the P-192 elliptic curve is used.

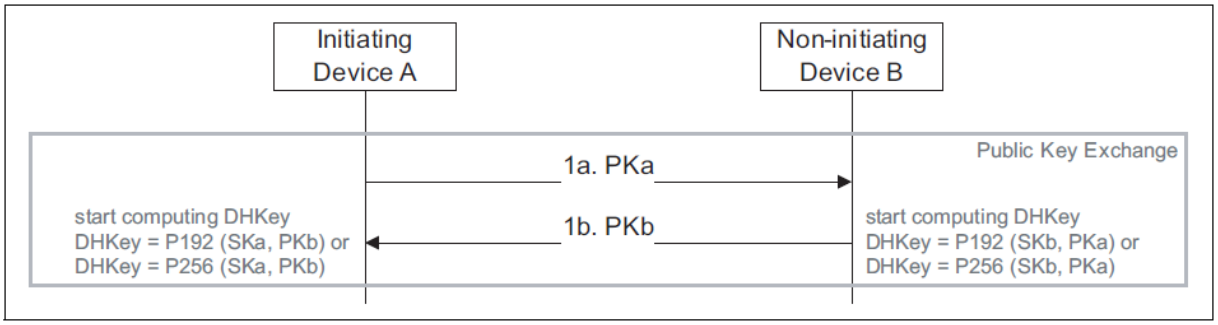


Figure 7.2: Public Key Exchange Details

A device shall validate that any public key received from any BD_ADDR is on the correct curve (P-192 or P-256) – see Section 7.6.

4.3.3 [Modified Section] 7.6 ELLIPTIC CURVE DEFINITION

[The modified text with changes is shown below. Two new paragraphs have been inserted at the end.]

For P-256:

```
p = 11579208921035624876269744694940757353008614341529031419553363130
8867097853951
r = 11579208921035624876269744694940757352999695522413576034242225906
1068512044369
```

```
b = 5ac635d8 aa3a93e7 b3ebbd55 769886bc 651d06b0 cc53b0f6 3bce3c3e
27d2604b
Gx = 6b17d1f2 e12c4247 f8bce6e5 63a440f2 77037d81 2deb33a0 f4a13945
d898c296
Gy = 4fe342e2 fe1a7f9b 8ee7eb4a 7c0f9e16 2bce3357 6b315ece cbb64068
37bf51f5
```

The function P-256 is defined as follows. Given an integer u, 0 < u < r, and a point V on the curve E, the value P-256(u,V) is computed as the x-coordinate of the u^{th} multiple uV of the point V.

The private keys shall be between 1 and r/2, where r is the Order of the Abelian Group on the elliptic curve (e.g. between 1 and $2^{256}/2$).

A valid public key Q = (X_Q, Y_Q) is one where X_Q and Y_Q are both in the range 0 to p - 1 and satisfy the equation $(Y_Q)^2 = (X_Q)^3 + aX_Q + b \pmod p$ in the relevant curve's finite field.

A device can validate a public key by directly checking the curve equation, by implementing elliptic curve point addition and doubling using formulas that are valid only on the correct curve, or by other means.

4.4 Changes to Core Specification v4.2, Volume 3, Part C: Generic Access Profile

4.4.1 **[Modified Section]** 14.1 CROSS-TRANSPORT KEY DERIVATION

[The modified section with changes is shown below.]

If both the local and remote devices support Secure Connections over the BR/EDR and LE transports, devices may optionally generate keys of identical strength and the same MITM protection for both transports as part of a single pairing procedure.

If both the local and remote devices support Secure Connections over the LE transport but not over the BR/EDR transport, then the devices may optionally generate the BR/EDR keys of identical strength and the same MITM protection as the LE keys as part of the LE pairing procedure.

If Secure Connections pairing occurs first on the LE transport the procedures in [Vol 3] Part H, Section 2.3.5.7 may be used.

If Secure Connections pairing occurs first on the BR/EDR transport the procedures in [Vol 3] Part H, Section 2.3.5.7 may be used.

If the BR/EDR link key has been generated by a Controller that does not perform remote public key validation (see [Vol 2] Part H, Section 7.6), then the LE LTK should not be generated from such a BR/EDR link key using cross-transport key derivation.

Note: The Host can use the HCI_Read_Local_Simple_Pairing_Options command (see [Vol 2] Part E, Section 7.4.X) or vendor-specific methods to determine whether the Controller performs remote public key validation.

If the LE LTK has been generated using the HCI_LE_Generate_DHKey command (see [Vol 2] Part E, Section 7.8.37) by a Controller that does not perform remote public key validation (see [Vol 3] Part H,

Section 2.3.5.6.1), then the BR/EDR link key should not be generated from such an LE LTK using cross-transport key derivation.

Note: The Host can use the Remote Public Key Validation feature bit (see [Vol 6] Part B, Section 4.6) or vendor-specific methods to determine whether the HCI_LE_Generate_DHKey command performs the remote public key validation.

4.5 Changes to Core Specification v4.2, Volume 3, Part H: Security Manager Specification

4.5.1 [Modified Section] *2.3.5.6.1 Public Key Exchange*

[The modified section with changes is shown below.]

Initially, each device generates its own Elliptic Curve Diffie-Hellman (ECDH) public-private key pair (phase 1). The public-private key pair contains a private (secret) key, and a public key. The private keys of devices A and B are denoted as SKa and SKb respectively. The public keys of devices A and B ~~and~~ are denoted as PKa and PKb respectively. ~~This key pair needs to be generated only once per device and may be computed in advance of pairing. A device may, at any time, choose to discard its public-private key pair and generate a new one, although there is not a requirement to do so.~~ See Section 2.3.6 for recommendations on how frequently this key pair should be changed.

Pairing is initiated by the initiating device sending its public key to the receiving device (phase 1a). The responding device replies with its own public key (phase 1b) These public keys are not regarded as secret although they may identify the devices. Note that phases 1a and 1b are the same in all three protocols.

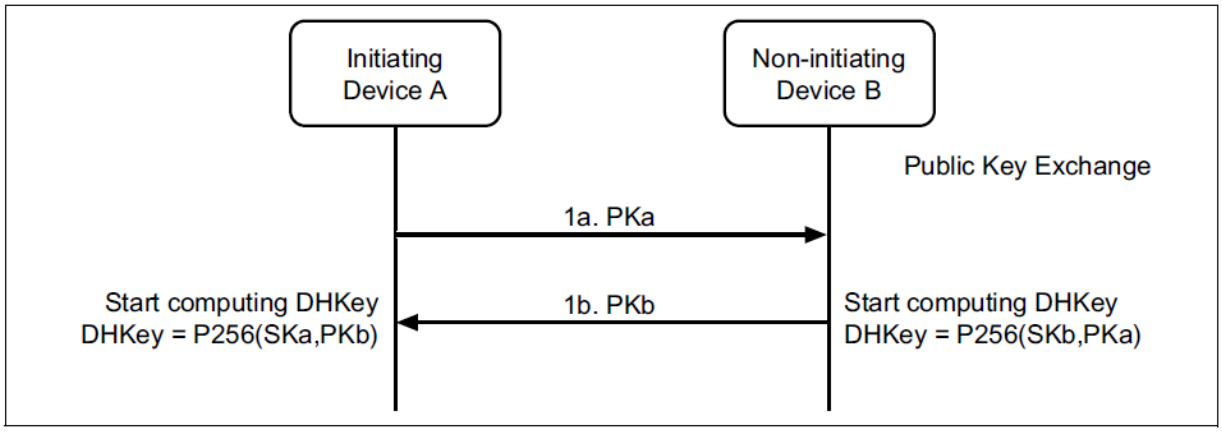


Figure 2.2: Public Key Exchange

A device shall validate that any public key received from any BD_ADDR is on the correct curve (P-256).

A valid public key Q = (X_Q, Y_Q) is one where X_Q and Y_Q are both in the range 0 to p - 1 and satisfy the equation $(Y_Q)^2 = (X_Q)^3 + aX_Q + b \pmod p$ in the relevant curve's finite field. See [Vol 2] Part H, Section 7.6 for the values of a, b, and p.

A device can validate a public key by directly checking the curve equation, by implementing elliptic curve point addition and doubling using formulas that are valid only on the correct curve, or by other means.

A device that detects an invalid public key from the peer at any point during the LE Secure Connections pairing process shall not use the resulting LTK, if any.

After the public keys have been exchanged, the device can then start computing the Diffie-Hellman Key.

When the Security Manager is placed in a Debug mode it shall use the following Diffie-Hellman private / public key pair:

Private key: `3f49f6d4 a3c55f38 74c9b3e3 d2103f50 4aff607b eb40b799 5899b8a6 cd3c1abd`
Public key (X): `20b003d2 f297be2c 5e2c83a7 e9f9a5b9 eff49111 acf4fddb cc030148 0e359de6`
Public key (Y): `dc809c49 652aeb6d 63329abf 5a52155c 766345c2 8fed3024 741c8ed0 1589d28b`

Note: Only one side (initiator or responder) needs to set Secure Connections debug mode in order for debug equipment to be able to determine the LTK and, therefore, be able to monitor the encrypted connection.

4.5.2 [Modified Section] 2.3.6 Repeated Attempts

[The modified section with changes is shown below.]

When a pairing procedure fails a waiting interval shall pass before the verifier will initiate a new Pairing Request command or Security Request command to the same claimant, or before it will respond to a Pairing Request command or Security Request command initiated by a device claiming the same identity as the failed device. For each subsequent failure, the waiting interval shall be increased exponentially. That is, after each failure, the waiting interval before a new attempt can be made, could be for example, twice as long as the waiting interval prior to the previous attempt[1]. The waiting interval should be limited to a maximum.

The maximum waiting interval depends on the implementation. The waiting time shall exponentially decrease to a minimum when no new failed attempts are made during a certain time period. This procedure prevents an intruder from repeating the pairing procedure with a large number of different keys.

To protect a device's private key, a device should implement a method to prevent an attacker from retrieving useful information about the device's private key ~~using invalid public keys~~. For this purpose, a device should change its private key after every pairing (successful or failed). Otherwise, it should change its private key at least after any of the following ~~can use one of the following methods~~:

- ~~Change its private key after~~ three failed attempts from any BD_ADDR ~~and~~
- ~~after 10~~ ten successful pairings from any BD_ADDR ~~; or~~
- ~~after~~ any combination of these such that ~~3~~ three successful pairings count as one failed pairing ~~.; or~~
- ~~Verify that the received public keys from any BD_ADDR are on the correct curve; or~~

[1] Another appropriate integer value larger than 1 may be used.

~~• Implement elliptic curve point addition and doubling using formulas that are valid only on the correct curve.~~

4.5.3 [Modified Section] 3.5.5 Pairing Failed

[The modified text with changes is shown below. A new paragraph has been inserted after the third paragraph and before Figure 3.7.]

This is used when there has been a failure during pairing and reports that the pairing procedure has been stopped and no further communication for the current pairing procedure is to occur. The Pairing Failed command is defined in Figure 3.7.

Any subsequent pairing procedure shall restart from the Pairing Feature Exchange phase.

This command may be sent at any time during the pairing process by either device in response to a message from the remote device.

During LE Secure Connections pairing, this command should be sent if the remote device's public key is invalid (see [Vol 3] Part H, Section 2.3.5.6.1). The Reason field should be set to "DHKey Check Failed".

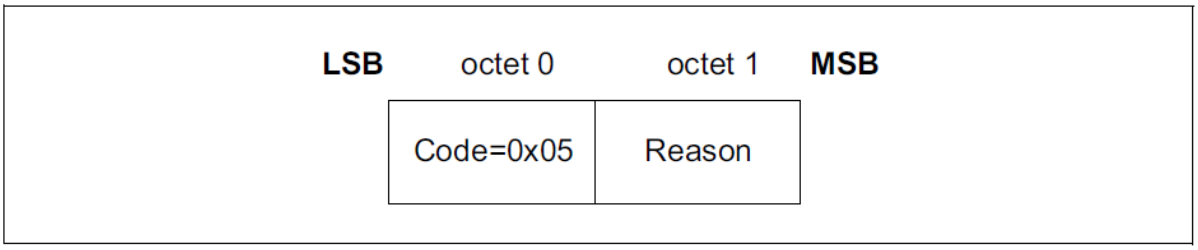


Figure 3.7: Pairing Failed Packet

4.6 Changes to Core Specification v4.2, Volume 6, Part B: Link Layer Specification

4.6.1 [Modified Section] 4.6 FEATURE SUPPORT

[The modified section with changes is shown below.]

When this information is sent from a Controller to a Host, a bit set to 0 indicates that the Link Layer Feature is not supported in this Controller; a bit set to 1 indicates that the Link Layer Feature is supported in this Controller.

When this information is sent from a Controller to a peer Controller, a bit set to 0 indicates that the Link Layer Feature shall not be used by the Controllers; a bit set to 1 indicates that the Link Layer Feature may be used by the Controllers.

The bit positions for each Link Layer Feature shall be as shown in Table 4.4. This table also shows if these bits are valid for the intended destination and which bits are masked before transmission to the peer device. If a bit is shown as not valid, using 'N', then this bit shall be ignored upon receipt. If a bit is

shown as masked, using 'Y', then this bit shall be set to zero when FeatureSet is sent to the peer device; otherwise, it shall be left unchanged.

Bit position	Link Layer Feature	Valid from Controller to Host	Valid from Controller to Controller	Masked to Peer
0	LE Encryption	Y	Y	N
1	Connection Parameters Request Procedure	Y	Y	N
2	Extended Reject Indication	Y	Y	N
3	Slave-initiated Features Exchange	Y	Y	N
4	LE Ping	Y	N	N
5	LE Data Packet Length Extension	Y	Y	N
6	LL Privacy	Y	N	N
7	Extended Scanner Filter Policies	Y	N	N
27	Remote Public Key Validation	Y	N	Y
~~8 – 63~~All other bits	RFU			

Table 4.4: FeatureSet field's bit mapping to Controller features

4.6.2 [New Section] 4.6.23 Remote Public Key Validation

[A new section is added as shown below.]

A Controller that supports Remote Public Key Validation shall validate the remote public key (see [Vol 3] Part H, Section 2.3.5.6.1) sent by the Host in the HCI_LE_Generate_DHKey command (see [Vol 2] Part E, Section 7.8.37).

4.6.3 [Modified Section] 5.1.4 Feature Exchange Procedure

[The modified section with changes is shown below.]

The Link Layer parameter for the current supported feature set (featureSet) may be exchanged after entering the Connection State. Both the master and slave can initiate this procedure.

The featureSet information may be cached. A Link Layer should not request this information on every connection if the information has been cached for this device. Cached information for a device may not be authoritative, and therefore an implementation must be able to accept the LL_UNKNOWN_RSP PDU if use of a feature is attempted that is not currently supported or used by the peer.

The featureSet$_M$ parameter is the feature capabilities of the Link Layer of the master with certain bits masked as specified in Section 4.6.

The featureSet$_S$ parameter is the feature capabilities of the Link Layer of the Slave with certain bits masked as specified in Section 4.6.

The featureSet$_{USED}$ is the logical AND of featureSet$_M$ and FeatureSet$_S$.

5 Changes to Core Specification v4.1

This Section sets forth the specific changes and additions, using the formatting and color conventions described in Section 2, to Core Specification v4.1.

5.1 Changes to Core Specification v4.1, Volume 2, Part C: Link Manager Protocol Specification

5.1.1 **[Modified Section]** 4.2.7.4 Authentication stage 2: DHKey Check

[The modified section with changes is shown below.]

At this stage, both devices compute new confirmation values based on Diffie-Hellman key and previously exchanged information according to [Vol 2] Part H, Section 7.7.4, “The Simple Pairing Check Function f3,” on page 1337.

The Initiator shall send an LMP_DHKey_check PDU to the Responder. If the Initiator has determined that the received public key is invalid (see [Vol 2] Part H, Section 7.6), the PDU should include a value that is different from the computed confirmation value (for example, substituting a randomly generated number). Otherwise, the PDU shall include ~~including~~ the computed confirmation value ~~it has computed~~.

[Insert a paragraph break.]

Upon reception, if the received value is not equal to the one calculated according to [Vol 2] Part H, Section 7.7.4, then the Responder shall follow the procedure in Section 4.2.7.4.1. If the values match, the Responder should follow the procedure in Section 4.2.7.4.1 if the received public key is invalid (see [Vol 2] Part H, Section 7.6). Otherwise, it shall reply with an LMP_accepted PDU ~~if the received value is equal to the one it has calculated according to Section 7.7.4 on page 1337. If it fails, refer to Section 4.2.7.3.5.1, “Check Failure on the Initiator Side,” on page 306~~.

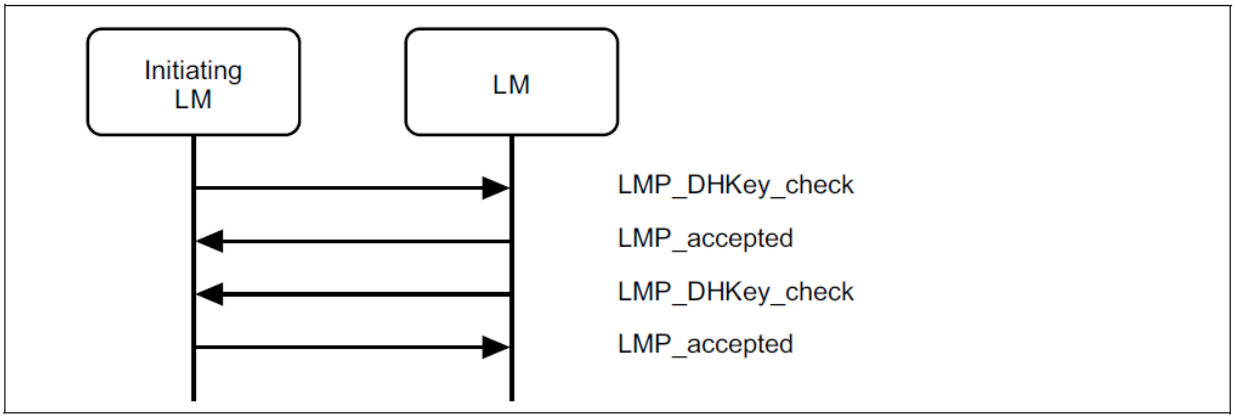
The Responder shall then send an LMP_DHKey_check PDU, ~~to the Initiator~~ including the ~~its~~ confirmation value it has computed, to the Initiator. Upon reception, if the received value is not equal to the one calculated according to [Vol 2] Part H, Section 7.7.4, then the Initiator shall follow the procedure in Section 4.2.7.4.1.1. If the values match, the Initiator should follow the procedure in Section 4.2.7.4.1.1 if the received public key is invalid (see [Vol 2] Part H, Section 7.6). Otherwise, it shall reply with an LMP_accepted PDU ~~if the received value is equal to the one it has calculated according to Section 7.7.4 on page 1337. If it fails, refer to Section 4.2.7.4.1.1, “Check Failure on the Initiator Side,” on page 309.~~

At this point, both devices shall compute the link key according to [Vol 2] Part H, Section 7.7.3, “The Simple Pairing Key Derivation Function f2,” on page 1336.

If at least one device does not support both the Secure Connections (Controller Support) and the Secure Connections (Host Support) features, the Initiator shall then start standard mutual authentication as described in Section 4.2.1.1

If both devices support both the Secure Connections (Controller Support) and the Secure Connections (Host Support) features, the Initiator shall then start secure authentication as described in Section 4.2.1.4.

After secure authentication, if encryption is enabled, the initiating device shall pause and immediately resume encryption to produce a new encryption key. Note: This will cause a new encryption key to be generated using the h3 function including the ACO created during the secure authentication process.



Sequence 71: DHKey check

A device that detects an invalid public key (see [Vol 2] Part H, Section 7.6) from the peer at any point during the Secure Simple Pairing process shall fail the pairing process and therefore not create a link key.

5.1.2 [Modified Section] *4.2.7.4.1 Check Failure on the Responder Side*

[The modified first paragraph with changes is shown below.]

If the received public key is invalid (see [Vol 2] Part H, Section 7.6), the Responder should send an LMP_not_accepted PDU with reason “Authentication Failure”. If the confirmation value received via LMP by the Responder is not equal to the one it has calculated according to [Vol 2] Part H, Section 7.7.4 on page 1337, the Responder shall send an LMP_not_accepted PDU with reason “Authentication Failure”.

5.1.3 [Modified Section] *4.2.7.4.1.1 Check Failure on the Initiator Side*

[The modified first paragraph with changes is shown below.]

If the received public key is invalid (see [Vol 2] Part H, Section 7.6), the Initiator should send an LMP_not_accepted PDU with reason “Authentication Failure”. If the confirmation value received via LMP by the Initiator is not equal to the one it has calculated according to [Vol 2] Part H, Section 7.7.4 on page 1337, the Initiator shall send an LMP_not_accepted PDU with reason “Authentication Failure”.

5.2 Changes to Core Specification v4.1, Volume 2, Part E: Host Controller Interface Functional Specification

5.2.1 [Modified Section] 3.4 CONTROLLER INFORMATION

[Insert a new row at the end of Table 3.5 as shown below.]

Name	Vers.	Summary description	Supported Controllers

Set MWS PATTERN Configuration Command	CSA3	The Set MWS PATTERN Command specifies the configuration of the pattern indicated over the MWS Coexistence Transport Layer.	BR/EDR, LE and AMP
Read Local Simple Pairing Options	Erratum 10734	The Read Local Simple Pairing Options command is used to read the simple pairing options and the maximum encryption key size supported.	BR/EDR

Table 3.5: Controller information

5.2.2 **[Modified Section]** 6.27 SUPPORTED COMMANDS

[The modified table with changes is shown below.]

Octet	Bit	Command Supported
33	0	Read Extended Page Timeout
	1	Write Extended Page Timeout
	2	Read Extended Inquiry Length
	3	Write Extended Inquiry Length
	4	LE Remote Connection Parameter Request Reply Command
	5	LE Remote Connection Parameter Request Negative Reply Command
	6	Reserved
	7	Reserved
34	All	Reserved
35	All	Reserved
36	All	Reserved
37	All	Reserved
38	All	Reserved
39	All	Reserved
40	All	Reserved
41	0	Reserved
	1	Reserved
	2	Reserved
	3	Read Local Simple Pairing Options
	4	Reserved
	5	Reserved
	6	Reserved
	7	Reserved

5.2.3 [New Section] 7.4.9 Read Local Simple Pairing Options Command

[A new section is added as shown below.]

Command	OCF	Command Parameters	Return Parameters
HCI_Read_Local_Simple_Pairing_Options	0x000C		Status, Simple_Pairing_Options, Maximum_Encryption_Key_Size

Description:

The Read_Local_Simple_Pairing_Options command is used to read the simple pairing options and the maximum encryption key size supported. Bit 0 of the Simple_Pairing_Options return parameter shall be set to 1.

Note: If this command is supported, then the Controller must support remote public key validation (see [Vol 2] Part H, Section 7.6).

Command parameters:

None

Return parameters:

Status: *Size: 1 octet*

Value	Parameter Description
0x00	Read_Local_Simple_Pairing_Options command succeeded.
0x01–0xFF	Read_Local_Simple_Pairing_Options command failed. See [Vol 2] Part D, Error Codes for a list of error codes and descriptions.

Simple_Pairing_Options: *Size: 1 octet*

Bit Number	Parameter Description
0	Remote public key validation is always performed.
All other bits	Reserved for future use.

Maximum_Encryption_Key_Size: *Size: 1 octet*

Value	Parameter Description
0x07–0x10	Maximum encryption key size (in octets) supported.
All other values	Reserved for future use.

Event(s) generated (unless masked away):

When the Read_Local_Simple_Pairing_Options command has completed, a Command Complete event shall be generated.

5.3 Changes to Core Specification v4.1, Volume 2, Part H: Security Specification

5.3.1 [Modified Section] 5.1 REPEATED ATTEMPTS

[The modified section with changes is shown below.]

When the authentication attempt fails, a waiting interval shall pass before the verifier will initiate a new authentication attempt to the same claimant, or before it will respond to an authentication attempt initiated by a device claiming the same identity as the failed device. For each subsequent authentication failure, the waiting interval shall be increased exponentially. For example, after each failure, the waiting interval before a new attempt can be made could be twice as long as the waiting interval prior to the previous attempt[1]. The waiting interval shall be limited to a maximum.

The maximum waiting interval depends on the implementation. The waiting time shall exponentially decrease to a minimum when no new failed attempts are made during a certain time period. This procedure prevents an intruder from repeating the authentication procedure with a large number of different keys.

To protect a device's private key, a device should implement a method to prevent an attacker from retrieving useful information about the device's private key ~~using invalid public keys~~. For this purpose, a device should change its private key after every pairing (successful or failed). Otherwise, it should change its private key at least after any of the following ~~can use one of the following methods~~:

- ~~Change its private key after~~ three failed attempts from any BD_ADDR ~~and~~
- ~~after 10~~ ten successful pairings from any BD_ADDR ~~; or~~
- ~~after~~ any combination of these such that ~~3~~ three successful pairings count as one failed pairing ~~.; or~~
- ~~Verify that the received public keys from any BD_ADDR are on the correct curve; or~~
- ~~Implement elliptic curve point addition and doubling using formulas that are valid only on the correct curve.~~

5.3.2 [Modified Section] 7.1 PHASE 1: PUBLIC KEY EXCHANGE

[The modified section with changes is shown below.]

Initially, each device generates its own Elliptic Curve Diffie-Hellman (ECDH) public-private key pair (step 1). ~~This key pair needs to be generated only once per device and may be computed in advance of pairing. A device may, at any time, choose to discard its public-private key pair and generate a new one, although there is not a requirement to do so.~~ See Section 5.1 for recommendations on how frequently this key pair should be changed.

Pairing is initiated by the initiating device sending its public key to the receiving device (step 1a). The responding device replies with its own public key (step 1b) These public keys are not regarded as secret although they may identify the devices. Note that steps 1a and 1b are the same in all three protocols.

When both device's Controllers and Hosts support Secure Connections, the P-256 elliptic curve is used. When at least one device's Controller or Host doesn't support Secure Connections, the P-192 elliptic curve is used.

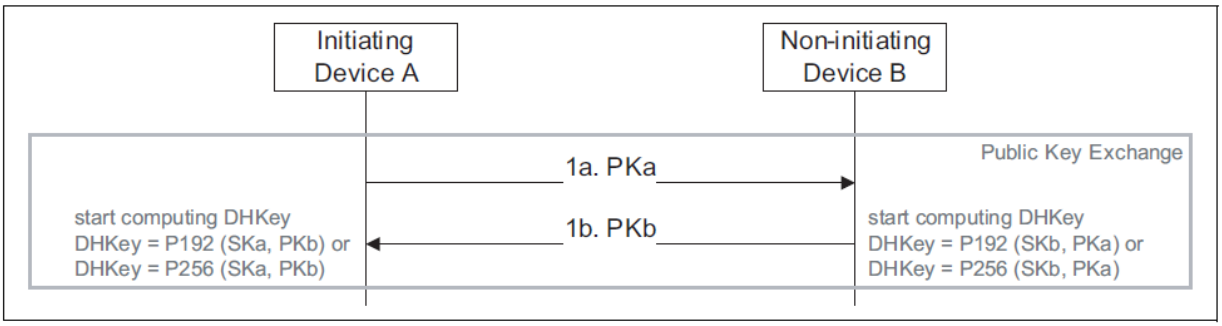


Figure 7.2: Public Key Exchange Details

A device shall validate that any public key received from any BD_ADDR is on the correct curve (P-192 or P-256) – see Section 7.6.

5.3.3 [Modified Section] 7.6 ELLIPTIC CURVE DEFINITION

[The modified text with changes is shown below. Two new paragraphs have been inserted at the end.]

For P-256:

```
p = 115792089210356248762697446949407573530086143415290314195533631308867097853951
r = 115792089210356248762697446949407573529996955224135760342422259061068512044369
b = 5ac635d8 aa3a93e7 b3ebbd55 769886bc 651d06b0 cc53b0f6 3bce3c3e 27d2604b
Gx = 6b17d1f2 e12c4247 f8bce6e5 63a440f2 77037d81 2deb33a0 f4a13945 d898c296
Gy = 4fe342e2 fe1a7f9b 8ee7eb4a 7c0f9e16 2bce3357 6b315ece cbb64068 37bf51f5
```

The function P-256 is defined as follows. Given an integer u, 0 < u < r, and a point V on the curve E, the value P-256(u,V) is computed as the x-coordinate of the u^{th} multiple uV of the point V.

The private keys shall be between 1 and r/2, where r is the Order of the Abelian Group on the elliptic curve (e.g. between 1 and $2^{256}/2$).

A valid public key Q = (X_Q, Y_Q) is one where X_Q and Y_Q are both in the range 0 to p - 1 and satisfy the equation $(Y_Q)^2 = (X_Q)^3 + aX_Q + b \pmod p$ in the relevant curve's finite field.

A device can validate a public key by directly checking the curve equation, by implementing elliptic curve point addition and doubling using formulas that are valid only on the correct curve, or by other means.

6 Changes to Core Specification v4.0

This Section sets forth the specific changes and additions, using the formatting and color conventions described in Section 2, to Core Specification v4.0.

6.1 Changes to Core Specification v4.0, Volume 2, Part C: Link Manager Protocol Specification

6.1.1 [Modified Section] *4.2.7.4 Authentication stage 2: DHKey Check*

[The modified section with changes is shown below.]

At this stage, both devices compute new confirmation values based on Diffie-Hellman key and previously exchanged information according to [Vol 2] Part H, Section 7.7.4, “The Simple Pairing Check Function f3,” on page 1105.

The Initiator shall send an LMP_DHKey_check PDU to the Responder. If the Initiator has determined that the received public key is invalid (see [Vol 2] Part H, Section 7.6), the PDU should include a value that is different from the computed confirmation value (for example, substituting a randomly generated number). Otherwise, the PDU shall include ~~including~~ the computed confirmation value ~~it has computed~~.

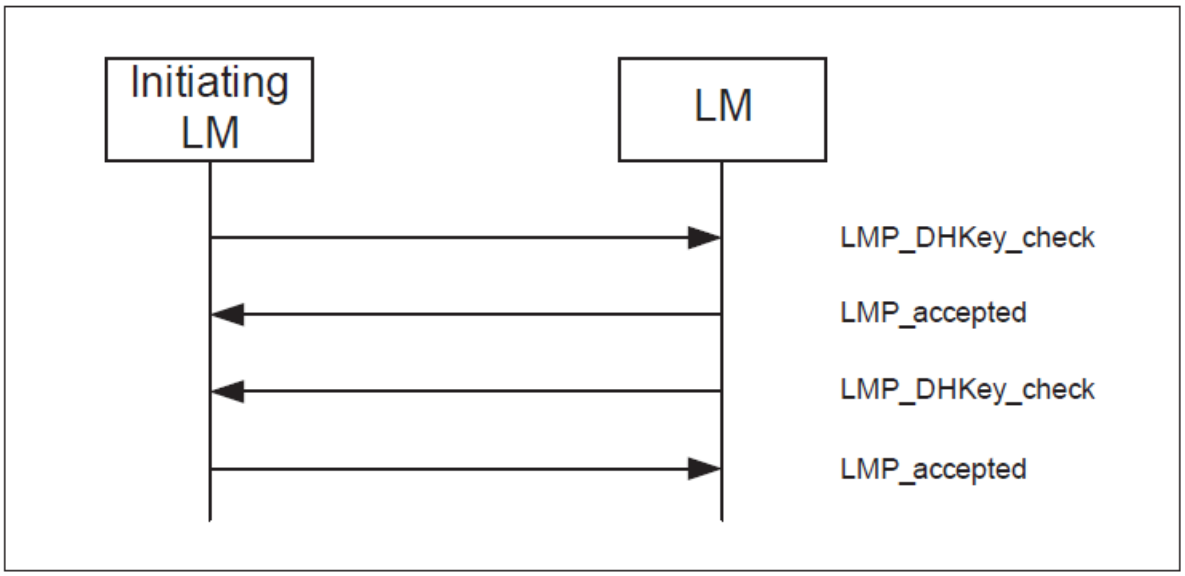
[Insert a paragraph break.]

Upon reception, if the received value is not equal to the one calculated according to [Vol 2] Part H, Section 7.7.4, then the Responder shall follow the procedure in Section 4.2.7.4.1. If the values match, the Responder should follow the procedure in Section 4.2.7.4.1 if the received public key is invalid (see [Vol 2] Part H, Section 7.6). Otherwise, it shall reply with an LMP_accepted PDU ~~if the received value is equal to the one it has calculated according to Section 7.7.4 on page 1105. If it fails, refer to Section 4.2.7.3.5.1, “Check Failure on the Initiator Side,” on page 272~~.

The Responder shall then send an LMP_DHKey_check PDU, ~~to the Initiator~~ including the ~~its~~ confirmation value it has computed, to the Initiator. Upon reception, if the received value is not equal to the one calculated according to [Vol 2] Part H, Section 7.7.4, then the Initiator shall follow the procedure in Section 4.2.7.4.1.1. If the values match, the Initiator should follow the procedure in Section 4.2.7.4.1.1 if the received public key is invalid (see [Vol 2] Part H, Section 7.6). Otherwise, it shall reply with an LMP_accepted PDU ~~if the received value is equal to the one it has calculated according to Section 7.7.4 on page 1105. If it fails, refer to Section 4.2.7.4.1.1, “Check Failure on the Initiator Side,” on page 275.~~

At this point, both devices shall compute the link key according to [Vol 2] Part H, Section 7.7.3, “The Simple Pairing Key Derivation Function f2,” on page 1104.

The Initiator shall then start standard mutual authentication as described in Section 4.2.1.1



Sequence 63: DHKey check

A device that detects an invalid public key (see [Vol 2] Part H, Section 7.6) from the peer at any point during the Secure Simple Pairing process shall fail the pairing process and therefore not create a link key.

6.1.2 [Modified Section] *4.2.7.4.1 Check Failure on the Responder Side*

[The modified first paragraph with changes is shown below.]

If the received public key is invalid (see [Vol 2] Part H, Section 7.6), the Responder should send an LMP_not_accepted PDU with reason “Authentication Failure”. If the confirmation value received via LMP by the Responder is not equal to the one it has calculated according to [Vol 2] Part H, Section 7.7.4 on page 1105, the Responder shall send an LMP_not_accepted PDU with reason “Authentication Failure”.

6.1.3 [Modified Section] *4.2.7.4.1.1 Check Failure on the Initiator Side*

[The modified first paragraph with changes is shown below.]

If the received public key is invalid (see [Vol 2] Part H, Section 7.6), the Initiator should send an LMP_not_accepted PDU with reason “Authentication Failure”. If the confirmation value received via LMP by the Initiator is not equal to the one it has calculated according to [Vol 2] Part H, Section 7.7.4 on page 1105, the Initiator shall send an LMP_not_accepted PDU with reason “Authentication Failure”.

6.2 Changes to Core Specification v4.0, Volume 2, Part E: Host Controller Interface Functional Specification

6.2.1 **[Modified Section]** 3.4 CONTROLLER INFORMATION

[Insert a new row at the end of Table 3.5 as shown below.]

Name	Vers.	Summary description	Supported Controllers
LE Read Supported States Command	4.0	The LE Read Supported States Command will read the current supported state and role combinations for the local LE Controllers.	LE
Read Local Simple Pairing Options	Erratum 10734	The Read Local Simple Pairing Options command is used to read the simple pairing options and the maximum encryption key size supported.	BR/EDR

Table 3.5: Controller information

6.2.2 **[Modified Section]** 6.27 SUPPORTED COMMANDS

[The modified table with changes is shown below. Note: CSA2 alters octet 29, CSA3 alters octets 29 and 30, and CSA4 alters octets 29 to 32. In each case the CSA change overrides the changes in this erratum.]

Octet	Bit	Command Supported
28	0	LE Start Encryption
	1	LE Long Term Key Request Reply
	2	LE Long Term Key Requested Negative Reply
	3	LE Read Supported States
	4	LE Receiver Test
	5	LE Transmitter Test
	6	LE Test End
	7	Reserved
29	All	Reserved
30	All	Reserved
31	All	Reserved
32	All	Reserved
33	All	Reserved
34	All	Reserved
35	All	Reserved
36	All	Reserved
37	All	Reserved

38	All	Reserved
39	All	Reserved
40	All	Reserved
41	0	Reserved
	1	Reserved
	2	Reserved
	3	Read Local Simple Pairing Options
	4	Reserved
	5	Reserved
	6	Reserved
	7	Reserved

6.2.3 [New Section] 7.4.9 Read Local Simple Pairing Options Command

[A new section is added as shown below.]

Command	OCF	Command Parameters	Return Parameters
HCI_Read_Local_Simple_Pairing_Options	0x000C		Status, Simple_Pairing_Options, Maximum_Encryption_Key_Size

Description:

The Read_Local_Simple_Pairing_Options command is used to read the simple pairing options and the maximum encryption key size supported. Bit 0 of the Simple_Pairing_Options return parameter shall be set to 1.

Note: If this command is supported, then the Controller must support remote public key validation (see [Vol 2] Part H, Section 7.6).

Command parameters:

None

Return parameters:

Status: *Size: 1 octet*

Value	Parameter Description
0x00	Read_Local_Simple_Pairing_Options command succeeded.
0x01–0xFF	Read_Local_Simple_Pairing_Options command failed. See [Vol 2] Part D, Error Codes for a list of error codes and descriptions.

Simple_Pairing_Options: *Size: 1 octet*

Bit Number	Parameter Description
0	Remote public key validation is always performed.
All other bits	Reserved for future use.

Maximum_Encryption_Key_Size: *Size: 1 octet*

Value	Parameter Description
0x07–0x10	Maximum encryption key size (in octets) supported.
All other values	Reserved for future use.

Event(s) generated (unless masked away):

When the Read_Local_Simple_Pairing_Options command has completed, a Command Complete event shall be generated.

6.3 Changes to Core Specification v4.0, Volume 2, Part H: Security Specification

6.3.1 [Modified Section] 5.1 REPEATED ATTEMPTS

[The modified section with changes is shown below.]

When the authentication attempt fails, a waiting interval shall pass before the verifier will initiate a new authentication attempt to the same claimant, or before it will respond to an authentication attempt initiated by a device claiming the same identity as the failed device. For each subsequent authentication failure, the waiting interval shall be increased exponentially. That is, after each failure, the waiting interval before a new attempt can be made, could be for example, twice as long as the waiting interval prior to the previous attempt[1]. The waiting interval shall be limited to a maximum.

The maximum waiting interval depends on the implementation. The waiting time shall exponentially decrease to a minimum when no new failed attempts are made during a certain time period. This procedure prevents an intruder from repeating the authentication procedure with a large number of different keys.

To protect a device's private key, a device should implement a method to prevent an attacker from retrieving useful information about the device's private key ~~using invalid public keys~~. For this purpose, a device should change its private key after every pairing (successful or failed). Otherwise, it should change its private key at least after any of the following ~~can use one of the following methods~~:

- ~~Change its private key after~~ three failed attempts from any BD_ADDR ~~and~~
- ~~after 10~~ ten successful pairings from any BD_ADDR ~~; or~~
- ~~after~~ any combination of these such that ~~3~~ three successful pairings count as one failed pairing ~~.; or~~
- ~~Verify that the received public keys from any BD_ADDR are on the correct curve; or~~

~~• Implement elliptic curve point addition and doubling using formulas that are valid only on the correct curve.~~

6.3.2 [Modified Section] 7.1 PHASE 1: PUBLIC KEY EXCHANGE

[The modified section with changes is shown below.]

Initially, each device generates its own Elliptic Curve Diffie-Hellman (ECDH) public-private key pair (step 1). ~~This key pair needs to be generated only once per device and may be computed in advance of pairing. A device may, at any time, choose to discard its public-private key pair and generate a new one, although there is not a requirement to do so.~~ See Section 5.1 for recommendations on how frequently this key pair should be changed.

Pairing is initiated by the initiating device sending its public key to the receiving device (step 1a). The responding device replies with its own public key (step 1b) These public keys are not regarded as secret although they may identify the devices. Note that steps 1a and 1b are the same in all three protocols.

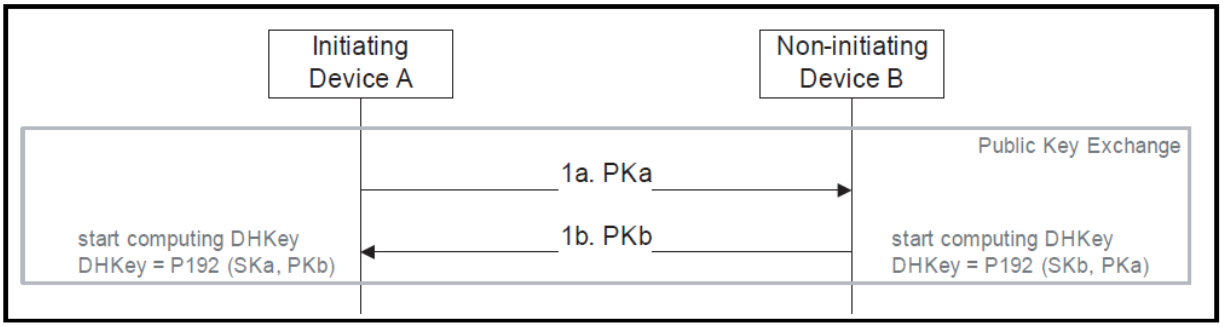


Figure 7.2: Public Key Exchange Details

A device shall validate that any public key received from any BD_ADDR is on the correct curve (P-192 or P-256) – see Section 7.6.

6.3.3 [Modified Section] 7.6 ELLIPTIC CURVE DEFINITION

[The modified text with changes is shown below. Two new paragraphs have been inserted at the end.]

The following parameters are given:

• The prime modulus p, order r, base point x-coordinate Gx, base point y-coordinate Gy.

• The integers p and r are given in decimal form; bit strings and field elements are given in hex.

```
p = 6277101735386680763835789423207666416083908700390324961279
r = 6277101735386680763835789423176059013767194773182842284081
b = 64210519 e59c80e7 0fa7e9ab 72243049 feb8deec c146b9b1
Gx = 188da80e b03090f6 7cbf20eb 43a18800 f4ff0afd 82ff1012
Gy = 07192b95 ffc8da78 631011ed 6b24cdd5 73f977a1 1e794811
```

The function P192 is defined as follows. Given an integer u, 0 < u < r, and a point V on the curve E, the value P192(u,V) is computed as the x-coordinate of the u^{th} multiple uV of the point V.

The private keys shall be between 1 and r/2, where r is the Order of the Abelian Group on the elliptic curve (e.g. between 1 and $2^{192}/2$).

A valid public key $Q = (X_Q, Y_Q)$ is one where X_Q and Y_Q are both in the range 0 to p - 1 and satisfy the equation $(Y_Q)^2 = (X_Q)^3 + aX_Q + b \pmod p$ in the relevant curve's finite field.

A device can validate a public key by directly checking the curve equation, by implementing elliptic curve point addition and doubling using formulas that are valid only on the correct curve, or by other means.

7 Changes to Core Specification v3.0 + HS

This Section sets forth the specific changes and additions, using the formatting and color conventions described in Section 2, to Core Specification v3.0 + HS.

7.1 Changes to Core Specification v3.0 + HS, Volume 2, Part C: Link Manager Protocol Specification

7.1.1 [Modified Section] *4.2.7.4 Authentication stage 2: DHKey Check*

[The modified section with changes is shown below.]

At this stage, both devices compute a new confirmation values based on Diffie-Hellman key and previously exchanged information according to [Vol 2] Part H, Section 7.7.4, “The Simple Pairing Check Function f3,” on page 967.

The Initiator shall send an LMP_DHKey_check PDU to the Responder. If the Initiator has determined that the received public key is invalid (see [Vol 2] Part H, Section 7.6), the PDU should include a value that is different from the computed confirmation value (for example, substituting a randomly generated number). Otherwise, the PDU shall include ~~including~~ the computed confirmation value ~~it has computed~~.

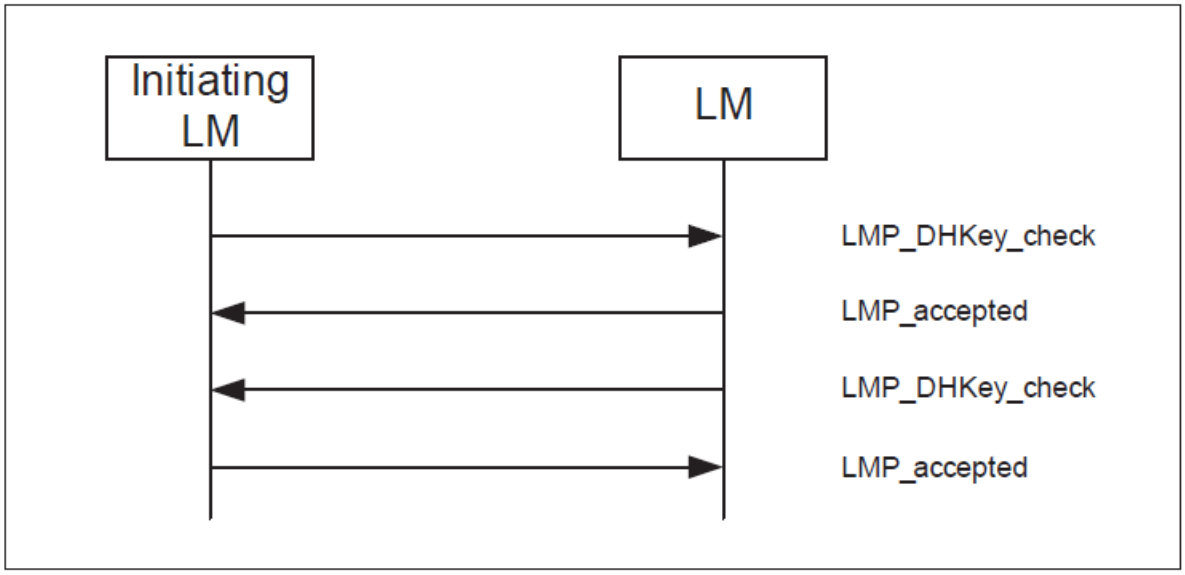
[Insert a paragraph break.]

Upon reception, if the received value is not equal to the one calculated according to [Vol 2] Part H, Section 7.7.4, then the Responder shall follow the procedure in Section 4.2.7.4.1. If the values match, the Responder should follow the procedure in Section 4.2.7.4.1 if the received public key is invalid (see [Vol 2] Part H, Section 7.6). Otherwise, it shall reply with an LMP_accepted PDU ~~if the received value is equal to the one it has calculated according to Section 7.7.4 on page 967. If it fails, refer to Section 4.2.7.3.5.1 on page 269~~.

The Responder shall then send an LMP_DHKey_check PDU, ~~to the Initiator~~ including the ~~its~~ confirmation value it has computed, to the Initiator. Upon reception, if the received value is not equal to the one calculated according to [Vol 2] Part H, Section 7.7.4, then the Initiator shall follow the procedure in Section 4.2.7.4.1.1. If the values match, the Initiator should follow the procedure in Section 4.2.7.4.1.1 if the received public key is invalid (see [Vol 2] Part H, Section 7.6). Otherwise, it shall reply with an LMP_accepted PDU ~~if the received value is equal to the one it has calculated according to Section 7.7.4 on page 967. If it fails, refer to Section 4.2.7.4.1.1, “Check Failure on the Initiator Side,” on page 272.~~

At this point, both devices shall compute the link key according to [Vol 2] Part H, Section 7.7.3, “The Simple Pairing Key Derivation Function f2,” on page 966.

The Initiator shall then start standard mutual authentication as described in Section 4.2.1.1



Sequence 63: DHKey check

A device that detects an invalid public key (see [Vol 2] Part H, Section 7.6) from the peer at any point during the Secure Simple Pairing process shall fail the pairing process and therefore not create a link key.

7.1.2 [Modified Section] *4.2.7.4.1 Check Failure on the Responder Side*

[The modified first paragraph with changes is shown below.]

If the received public key is invalid (see [Vol 2] Part H, Section 7.6), the Responder should send an LMP_not_accepted PDU with reason “Authentication Failure”. If the confirmation value received via LMP by the Responder is not equal to the one it has calculated according to [Vol 2] Part H, Section 7.7.4 on page 967, the Responder shall send an LMP_not_accepted PDU with reason “Authentication Failure”.

7.1.3 [Modified Section] *4.2.7.4.1.1 Check Failure on the Initiator Side*

[The modified first paragraph with changes is shown below.]

If the received public key is invalid (see [Vol 2] Part H, Section 7.6), the Initiator should send an LMP_not_accepted PDU with reason “Authentication Failure”. If the confirmation value received via LMP by the Initiator is not equal to the one it has calculated according to [Vol 2] Part H, Section 7.7.4 on page 967, the Initiator shall send an LMP_not_accepted PDU with reason “Authentication Failure”.

7.2 Changes to Core Specification v3.0 + HS, Volume 2, Part E: Host Controller Interface Functional Specification

7.2.1 **[Modified Section]** 3.4 CONTROLLER INFORMATION

Insert a new row at the end of Table 3.4 as shown below.]

Name	Vers.	Summary description	Supported Controllers
Read BD_ADDR Command	1.1	The Read BD_ADDR command will read the value for the BD_ADDR parameter.	BR/EDR
Read Local Simple Pairing Options	Erratum 10734	The Read Local Simple Pairing Options command is used to read the simple pairing options and the maximum encryption key size supported.	BR/EDR

Table 3.4: Controller information

7.2.2 **[Modified Section]** 6.26 SUPPORTED COMMANDS

[The modified table with changes is shown below. Note: CSA2 alters octet 29, CSA3 alters octets 29 and 30, and CSA4 alters octets 29 to 32. In each case the CSA change overrides the changes in this erratum.]

Octet	Bit	Command Supported
24	0	Read Enhanced Transmit Power Level
	1	Reserved
	2	Read Best Effort Flush Timeout
	3	Write Best Effort Flush Timeout
	4	Short Range Mode
	5	Reserved
	6	Reserved
	7	Reserved
25	All	Reserved
26	All	Reserved
27	All	Reserved
28	All	Reserved
29	All	Reserved
30	All	Reserved
31	All	Reserved
32	All	Reserved
33	All	Reserved
34	All	Reserved
35	All	Reserved

36	All	Reserved
37	All	Reserved
38	All	Reserved
39	All	Reserved
40	All	Reserved
41	0	Reserved
	1	Reserved
	2	Reserved
	3	Read Local Simple Pairing Options
	4	Reserved
	5	Reserved
	6	Reserved
	7	Reserved

7.2.3 [New Section] 7.4.9 Read Local Simple Pairing Options Command

[A new section is added as shown below.]

Command	OCF	Command Parameters	Return Parameters
HCI_Read_Local_Simple_Pairing_Options	0x000C		Status, Simple_Pairing_Options, Maximum_Encryption_Key_Size

Description:

The Read_Local_Simple_Pairing_Options command is used to read the simple pairing options and the maximum encryption key size supported. Bit 0 of the Simple_Pairing_Options return parameter shall be set to 1.

Note: If this command is supported, then the Controller must support remote public key validation (see [Vol 2] Part H, Section 7.6).

Command parameters:

None

Return parameters:

Status: *Size: 1 octet*

Value	Parameter Description
0x00	Read_Local_Simple_Pairing_Options command succeeded.
0x01–0xFF	Read_Local_Simple_Pairing_Options command failed. See [Vol 2] Part D, Error Codes for a list of error codes and descriptions.

Simple_Pairing_Options: *Size: 1 octet*

Bit Number	Parameter Description
0	Remote public key validation is always performed.
All other bits	Reserved for future use.

Maximum_Encryption_Key_Size: *Size: 1 octet*

Value	Parameter Description
0x07–0x10	Maximum encryption key size (in octets) supported.
All other values	Reserved for future use.

Event(s) generated (unless masked away):

When the Read_Local_Simple_Pairing_Options command has completed, a Command Complete event shall be generated.

7.3 Changes to Core Specification v3.0 + HS, Volume 2, Part H: Security Specification

7.3.1 [Modified Section] 5.1 REPEATED ATTEMPTS

[The modified section with changes is shown below.]

When the authentication attempt fails, a waiting interval shall pass before the verifier will initiate a new authentication attempt to the same claimant, or before it will respond to an authentication attempt initiated by a device claiming the same identity as the failed device. For each subsequent authentication failure, the waiting interval shall be increased exponentially. That is, after each failure, the waiting interval before a new attempt can be made, could be for example, twice as long as the waiting interval prior to the previous attempt[1]. The waiting interval shall be limited to a maximum.

The maximum waiting interval depends on the implementation. The waiting time shall exponentially decrease to a minimum when no new failed attempts are made during a certain time period. This procedure prevents an intruder from repeating the authentication procedure with a large number of different keys.

To protect a device's private key, a device should implement a method to prevent an attacker from retrieving useful information about the device's private key ~~using invalid public keys~~. For this purpose, a device should change its private key after every pairing (successful or failed). Otherwise, it should change its private key at least after any of the following ~~can use one of the following methods~~:

- ~~Change its private key after~~ three failed attempts from any BD_ADDR ~~and~~
- ~~after 10~~ ten successful pairings from any BD_ADDR ~~; or~~
- ~~after~~ any combination of these such that ~~3~~ three successful pairings count as one failed pairing ~~.; or~~
- ~~Verify that the received public keys from any BD_ADDR are on the correct curve; or~~

~~• Implement elliptic curve point addition and doubling using formulas that are valid only on the correct curve.~~

7.3.2 [Modified Section] 7.1 PHASE 1: PUBLIC KEY EXCHANGE

[The modified section with changes is shown below.]

Initially, each device generates its own Elliptic Curve Diffie-Hellman (ECDH) public-private key pair (step 1). ~~This key pair needs to be generated only once per device and may be computed in advance of pairing. A device may, at any time, choose to discard its public-private key pair and generate a new one, although there is not a requirement to do so.~~ See Section 5.1 for recommendations on how frequently this key pair should be changed.

Pairing is initiated by the initiating device sending its public key to the receiving device (step 1a). The responding device replies with its own public key (step 1b) These public keys are not regarded as secret although they may identify the devices. Note that steps 1a and 1b are the same in all three protocols.

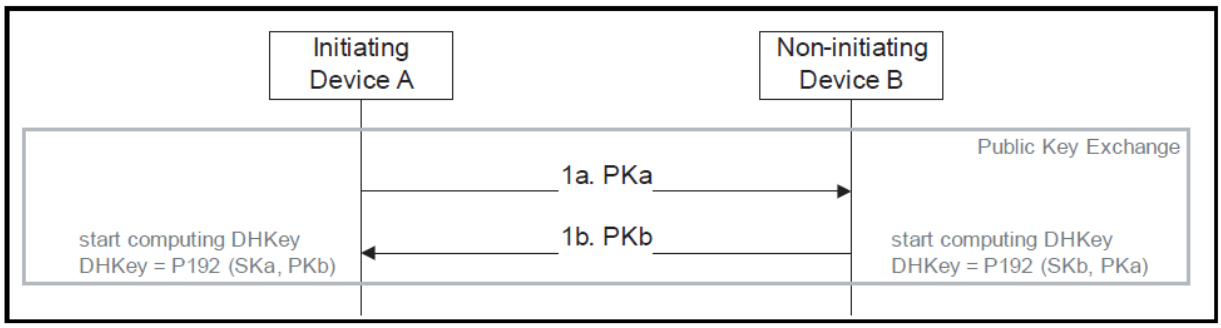


Figure 7.2: Public Key Exchange Details

A device shall validate that any public key received from any BD_ADDR is on the correct curve (P-192 or P-256) – see Section 7.6.

7.3.3 [Modified Section] 7.6 ELLIPTIC CURVE DEFINITION

[The modified text with changes is shown below. Two new paragraphs have been inserted at the end.]

The following parameters are given:

• The prime modulus p, order r, base point x-coordinate Gx, base point y-coordinate Gy.

• The integers p and r are given in decimal form; bit strings and field elements are given in hex.

```
p = 6277101735386680763835789423207666416083908700390324961279
r = 6277101735386680763835789423176059013767194773182842284081
b = 64210519 e59c80e7 0fa7e9ab 72243049 feb8deec c146b9b1
Gx = 188da80e b03090f6 7cbf20eb 43a18800 f4ff0afd 82ff1012
Gy = 07192b95 ffc8da78 631011ed 6b24cdd5 73f977a1 1e794811
```

The function P192 is defined as follows. Given an integer u, 0 < u < r, and a point V on the curve E, the value P192(u,V) is computed as the x-coordinate of the u^{th} multiple uV of the point V.

The private keys shall be between 1 and r/2, where r is the Order of the Abelian Group on the elliptic curve (e.g. between 1 and $2^{192}/2$).

A valid public key $Q = (X_Q, Y_Q)$ is one where X_Q and Y_Q are both in the range 0 to p - 1 and satisfy the equation $(Y_Q)^2 = (X_Q)^3 + aX_Q + b \pmod p$ in the relevant curve's finite field.

A device can validate a public key by directly checking the curve equation, by implementing elliptic curve point addition and doubling using formulas that are valid only on the correct curve, or by other means.

8 Changes to Core Specification v2.1 + EDR

This Section sets forth the specific changes and additions, using the formatting and color conventions described in Section 2, to Core Specification v2.1 + EDR.

8.1 Changes to Core Specification v2.1 + EDR, Volume 2, Part C: Link Manager Protocol Specification

8.1.1 [Modified Section] *4.2.7.4 Authentication stage 2: DHKey Check*

[The modified section with changes is shown below.]

At this stage, both devices compute a new confirmation values based on Diffie-Hellman key and previously exchanged information according to [Vol 2] Part H, Section 7.7.4, “The Simple Pairing Check Function f3,” on page 900.

The Initiator shall send an LMP_DHKey_check PDU to the Responder. If the Initiator has determined that the received public key is invalid (see [Vol 2] Part H, Section 7.6), the PDU should include a value that is different from the computed confirmation value (for example, substituting a randomly generated number). Otherwise, the PDU shall include ~~including~~ the computed confirmation value ~~it has computed~~.

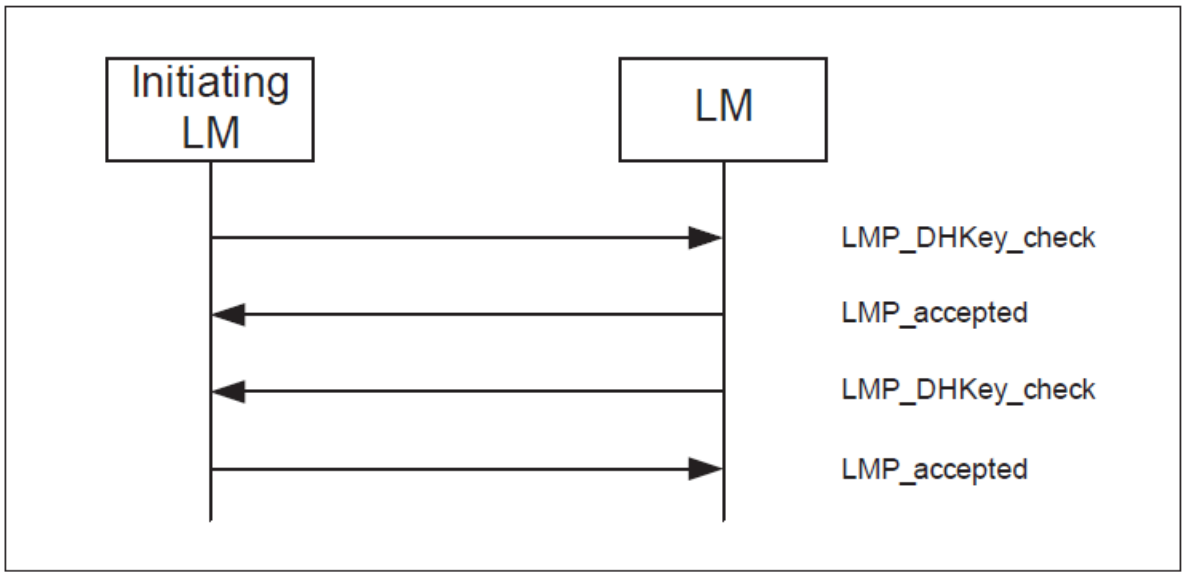
[Insert a paragraph break.]

Upon reception, if the received value is not equal to the one calculated according to [Vol 2] Part H, Section 7.7.4, then the Responder shall follow the procedure in Section 4.2.7.4.1. If the values match, the Responder should follow the procedure in Section 4.2.7.4.1 if the received public key is invalid (see [Vol 2] Part H, Section 7.6). Otherwise, it shall reply with an LMP_accepted PDU ~~if the received value is equal to the one it has calculated according to Section 7.7.4 on page 900. If it fails, refer to Section 4.2.7.3.5.1, “Check Failure on the Initiator Side,” on page 265~~.

The Responder shall then send an LMP_DHKey_check PDU, ~~to the Initiator~~ including the ~~its~~ confirmation value it has computed, to the Initiator. Upon reception, if the received value is not equal to the one calculated according to [Vol 2] Part H, Section 7.7.4, then the Initiator shall follow the procedure in Section 4.2.7.4.1.1. If the values match, the Initiator should follow the procedure in Section 4.2.7.4.1.1 if the received public key is invalid (see [Vol 2] Part H, Section 7.6). Otherwise, it shall reply with an LMP_accepted PDU ~~if the received value is equal to the one it has calculated according to Section 7.7.4 on page 900. If it fails, refer to Section 4.2.7.4.1.1, “Check failure on the initiator side,” on page 268.~~

At this point, both devices shall compute the link key according to [Vol 2] Part H, Section 7.7.3, “The Simple Pairing Key Derivation Function f2,” on page 899.

The Initiator shall then start standard mutual authentication as described in Section 4.2.1.1



Sequence 61: DHKey check

A device that detects an invalid public key (see [Vol 2] Part H, Section 7.6) from the peer at any point during the Secure Simple Pairing process shall fail the pairing process and therefore not create a link key.

8.1.2 [Modified Section] *4.2.7.4.1 Check failure on the Responder side*

[The modified first paragraph with changes is shown below.]

If the received public key is invalid (see [Vol 2] Part H, Section 7.6), the Responder should send an LMP_not_accepted PDU with reason “Authentication Failure”. If the confirmation value received via LMP by the Responder is not equal to the one it has calculated according to [Vol 2] Part H, Section 7.7.4 on page 900, the Responder shall send an LMP_not_accepted PDU with reason “Authentication Failure”.

8.1.3 [Modified Section] *4.2.7.4.1.1 Check failure on the initiator side*

[The modified first paragraph with changes is shown below.]

If the received public key is invalid (see [Vol 2] Part H, Section 7.6), the Initiator should send an LMP_not_accepted PDU with reason “Authentication Failure”. If the confirmation value received via LMP by the Initiator is not equal to the one it has calculated according to [Vol 2] Part H, Section 7.7.4 on page 900, the Initiator shall send an LMP_not_accepted PDU with reason “Authentication Failure”.

8.2 Changes to Core Specification v2.1 + EDR, Volume 2, Part E: Host Controller Interface Functional Specification

8.2.1 **[Modified Section]** 3.4 CONTROLLER INFORMATION

[Insert a new row at the end of Table 3.4 as shown below.]

Name	Vers.	Summary description
Read BD_ADDR Command	1.1	The Read BD_ADDR command will read the value for the BD_ADDR parameter.
Read Local Simple Pairing Options	Erratum 10734	The Read Local Simple Pairing Options command is used to read the simple pairing options and the maximum encryption key size supported.

Table 3.4: Controller information

8.2.2 **[Modified Section]** 6.26 SUPPORTED COMMANDS

[The modified table with changes is shown below. Note: CSA2 alters octet 29 and CSA3 alters octets 29 and 30. In each case the CSA change overrides the changes in this erratum.]

Octet	Bit	Command Supported
20	0	Reserved
	1	Reserved
	2	Send Keypress Notification
	3	IO Capabilities Response Negative Reply
	4	Reserved
	5	Reserved
	6	Reserved
	7	Reserved
21	All	Reserved
22	All	Reserved
23	All	Reserved
24	All	Reserved
25	All	Reserved
26	All	Reserved
27	All	Reserved
28	All	Reserved
29	All	Reserved
30	All	Reserved
31	All	Reserved
32	All	Reserved
33	All	Reserved

34	All	Reserved
35	All	Reserved
36	All	Reserved
37	All	Reserved
38	All	Reserved
39	All	Reserved
40	All	Reserved
41	0	Reserved
	1	Reserved
	2	Reserved
	3	Read Local Simple Pairing Options
	4	Reserved
	5	Reserved
	6	Reserved
	7	Reserved

8.2.3 [New Section] 7.4.9 Read Local Simple Pairing Options Command

[A new section is added as shown below.]

Command	OCF	Command Parameters	Return Parameters
HCI_Read_Local_Simple_Pairing_Options	0x000C		Status, Simple_Pairing_Options, Maximum_Encryption_Key_Size

Description:

The Read_Local_Simple_Pairing_Options command is used to read the simple pairing options and the maximum encryption key size supported. Bit 0 of the Simple_Pairing_Options return parameter shall be set to 1.

Note: If this command is supported, then the Controller must support remote public key validation (see [Vol 2] Part H, Section 7.6).

Command parameters:

None

Return parameters:

Status: *Size: 1 octet*

Value	Parameter Description
0x00	Read_Local_Simple_Pairing_Options command succeeded.
0x01–0xFF	Read_Local_Simple_Pairing_Options command failed. See [Vol 2] Part D, Error Codes for a list of error codes and descriptions.

Simple_Pairing_Options: *Size: 1 octet*

Bit Number	Parameter Description
0	Remote public key validation is always performed.
All other bits	Reserved for future use.

Maximum_Encryption_Key_Size: *Size: 1 octet*

Value	Parameter Description
0x07–0x10	Maximum encryption key size (in octets) supported.
All other values	Reserved for future use.

Event(s) generated (unless masked away):

When the Read_Local_Simple_Pairing_Options command has completed, a Command Complete event shall be generated.

8.3 Changes to Core Specification v2.1 + EDR, Volume 2, Part H: Security Specification

8.3.1 [Modified Section] 5.1 REPEATED ATTEMPTS

[The modified section with changes is shown below.]

When the authentication attempt fails, a waiting interval shall pass before the verifier will initiate a new authentication attempt to the same claimant, or before it will respond to an authentication attempt initiated by a device claiming the same identity as the failed device. For each subsequent authentication failure, the waiting interval shall be increased exponentially. That is, after each failure, the waiting interval before a new attempt can be made, could be for example, twice as long as the waiting interval prior to the previous attempt[1]. The waiting interval shall be limited to a maximum.

The maximum waiting interval depends on the implementation. The waiting time shall exponentially decrease to a minimum when no new failed attempts are made during a certain time period. This procedure prevents an intruder from repeating the authentication procedure with a large number of different keys.

To protect a device's private key, a device should implement a method to prevent an attacker from retrieving useful information about the device's private key ~~using invalid public keys~~. For this purpose, a

device should change its private key after every pairing (successful or failed). Otherwise, it should change its private key at least after any of the following ~~can use one of the following methods~~:

• ~~Change its private key after~~ three failed attempts from any BD_ADDR ~~and~~

• ~~after 10~~ ten successful pairings from any BD_ADDR~~; or~~

• ~~after~~ any combination of these such that ~~3~~ three successful pairings count as one failed pairing~~.; or~~

~~• Verify that the received public keys from any BD_ADDR are on the correct curve; or~~

~~• Implement elliptic curve point addition and doubling using formulas that are valid only on the correct curve.~~

8.3.2 [Modified Section] 7.1 PHASE 1: PUBLIC KEY EXCHANGE

[The modified section with changes is shown below.]

Initially, each device generates its own Elliptic Curve Diffie-Hellman (ECDH) public-private key pair (step 1). ~~This key pair needs to be generated only once per device and may be computed in advance of pairing. A device may, at any time, choose to discard its public-private key pair and generate a new one, although there is not a requirement to do so.~~ See Section 5.1 for recommendations on how frequently this key pair should be changed.

Pairing is initiated by the initiating device sending its public key to the receiving device (step 1a). The responding device replies with its own public key (step 1b) These public keys are not regarded as secret although they may identify the devices. Note that steps 1a and 1b are the same in all three protocols.

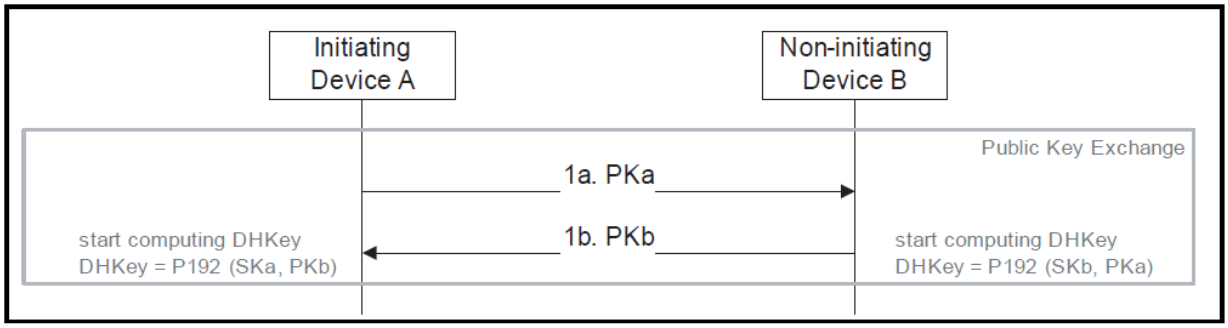


Figure 7.2: Public Key Exchange Details

A device shall validate that any public key received from any BD_ADDR is on the correct curve (P-192 or P-256) – see Section 7.6.

8.3.3 [Modified Section] 7.6 ELLIPTIC CURVE DEFINITION

[The modified text with changes is shown below. Two new paragraphs have been inserted at the end.]

The following parameters are given:

• The prime modulus p, order r, base point x-coordinate Gx, base point y-coordinate Gy.

• The integers p and r are given in decimal form; bit strings and field elements are given in hex.

```
p = 6277101735386680763835789423207666416083908700390324961279
r = 6277101735386680763835789423176059013767194773182842284081
b = 64210519 e59c80e7 0fa7e9ab 72243049 feb8deec c146b9b1
Gx = 188da80e b03090f6 7cbf20eb 43a18800 f4ff0afd 82ff1012
Gy = 07192b95 ffc8da78 631011ed 6b24cdd5 73f977a1 1e794811
```

The function P192 is defined as follows. Given an integer u, 0 < u < r, and a point V on the curve E, the value P192(u,V) is computed as the x-coordinate of the u^{th} multiple uV of the point V.

The private keys shall be between 1 and r/2, where r is the Order of the Abelian Group on the elliptic curve (e.g. between 1 and $2^{192}/2$).

A valid public key $Q = (X_Q, Y_Q)$ is one where X_Q and Y_Q are both in the range 0 to p - 1 and satisfy the equation $(Y_Q)^2 = (X_Q)^3 + aX_Q + b \pmod p$ in the relevant curve's finite field.

A device can validate a public key by directly checking the curve equation, by implementing elliptic curve point addition and doubling using formulas that are valid only on the correct curve, or by other means.

9 Missing section numbers

Where this erratum adds new sections to the Source Specifications, these new sections are given the same section number in all versions of the Source Specification (including versions under development and not yet published). This means that, in some cases, there will be section numbers missing in older Source Specification versions. For each such missing number, a new section is inserted with the title and content both “This section is not currently used”. The missing section numbers are as follows:

Source Specifications	Volume and Part	Missing section numbers
v2.1 + EDR	Volume 2, Part E	Sections 7.4.7 and 7.4.8
v3.0 + HS and v4.0	Volume 2, Part E	Section 7.4.8
v4.2	Volume 6, Part B	Sections 4.6.9 to 4.6.22
v5.0	Volume 6, Part B	Sections 4.6.16 to 4.6.22

10 References

[1] Core Specification version 5.0, dated 2016-Dec-06, location
https://www.bluetooth.org/DocMan/handlers/DownloadDoc.ashx?doc_id=421043

[2] Core Specification version 4.2, dated 2014-Dec-02, location
https://www.bluetooth.org/DocMan/handlers/DownloadDoc.ashx?doc_id=286439

[3] Core Specification version 4.1, dated 2013-Dec-03, location
https://www.bluetooth.org/DocMan/handlers/DownloadDoc.ashx?doc_id=282159

[4] Core Specification version 4.0, dated 2010-Jun-30, location
https://www.bluetooth.org/docman/handlers/downloaddoc.ashx?doc_id=229737

[5] Core Specification version 3.0 + HS, dated 2009-Apr-21, location
https://www.bluetooth.org/DocMan/handlers/DownloadDoc.ashx?doc_id=174214

[6] Core Specification version 2.1 + EDR, dated 2007-Jul-26, location
https://www.bluetooth.org/docman/handlers/downloaddoc.ashx?doc_id=241363