

# SPECIFICATION OF THE *BLUETOOTH* SYSTEM

Experience More



## Master Table of Contents & Compliance Requirements



Covered Core Package version:  
4.0  
Current Master TOC  
Publication date: 30 June 2010







## Revision History

The Revision History is shown in the [“Appendix” on page 55\[Vol. 0\]](#).

## Contributors

The persons who contributed to this specification are listed in the [Appendix](#).

## Web Site

This specification can also be found on the official Bluetooth web site:  
<http://www.bluetooth.com>

## Disclaimer and Copyright Notice

The copyright in these specifications is owned by the Promoter Members of Bluetooth SIG, Inc. (“Bluetooth SIG”). Use of these specifications and any related intellectual property (collectively, the “Specification”), is governed by the Promoters Membership Agreement among the Promoter Members and Bluetooth SIG (the “Promoters Agreement”), certain membership agreements between Bluetooth SIG and its Adopter and Associate Members (the “Membership Agreements”) and the Bluetooth Specification Early Adopters Agreements (“1.2 Early Adopters Agreements”) among Early Adopter members of the unincorporated Bluetooth special interest group and the Promoter Members (the “Early Adopters Agreement”). Certain rights and obligations of the Promoter Members under the Early Adopters Agreements have been assigned to Bluetooth SIG by the Promoter Members.

Use of the Specification by anyone who is not a member of Bluetooth SIG or a party to an Early Adopters Agreement (each such person or party, a “Member”), is prohibited. The legal rights and obligations of each Member are governed by their applicable Membership Agreement, Early Adopters Agreement or Promoters Agreement. No license, express or implied, by estoppel or otherwise, to any intellectual property rights are granted herein.

Any use of the Specification not in compliance with the terms of the applicable Membership Agreement, Early Adopters Agreement or Promoters Agreement is prohibited and any such prohibited use may result in termination of the applicable Membership Agreement or Early Adopters Agreement and other liability permitted by the applicable agreement or by applicable law to Bluetooth SIG or any of its members for patent, copyright and/or trademark infringement.



THE SPECIFICATION IS PROVIDED “AS IS” WITH NO WARRANTIES WHATSOEVER, INCLUDING ANY WARRANTY OF MERCHANTABILITY, NONINFRINGEMENT, FITNESS FOR ANY PARTICULAR PURPOSE, SATISFACTORY QUALITY, OR REASONABLE SKILL OR CARE, OR ANY WARRANTY ARISING OUT OF ANY COURSE OF DEALING, USAGE, TRADE PRACTICE, PROPOSAL, SPECIFICATION OR SAMPLE.

Each Member hereby acknowledges that products equipped with the Bluetooth® technology (“Bluetooth® Products”) may be subject to various regulatory controls under the laws and regulations of various governments worldwide. Such laws and regulatory controls may govern, among other things, the combination, operation, use, implementation and distribution of Bluetooth® Products. Examples of such laws and regulatory controls include, but are not limited to, airline regulatory controls, telecommunications regulations, technology transfer controls and health and safety regulations. Each Member is solely responsible for the compliance by their Bluetooth® Products with any such laws and regulations and for obtaining any and all required authorizations, permits, or licenses for their Bluetooth® Products related to such regulations within the applicable jurisdictions. Each Member acknowledges that nothing in the Specification provides any information or assistance in connection with securing such compliance, authorizations or licenses. NOTHING IN THE SPECIFICATION CREATES ANY WARRANTIES, EITHER EXPRESS OR IMPLIED, REGARDING SUCH LAWS OR REGULATIONS.

ALL LIABILITY, INCLUDING LIABILITY FOR INFRINGEMENT OF ANY INTELLECTUAL PROPERTY RIGHTS OR FOR NONCOMPLIANCE WITH LAWS, RELATING TO USE OF THE SPECIFICATION IS EXPRESSLY DISCLAIMED. BY USE OF THE SPECIFICATION, EACH MEMBER EXPRESSLY WAIVES ANY CLAIM AGAINST BLUETOOTH SIG AND ITS PROMOTER MEMBERS RELATED TO USE OF THE SPECIFICATION.

Bluetooth SIG reserves the right to adopt any changes or alterations to the Specification as it deems necessary or appropriate.

Copyright © 1999 - 2010

Ericsson AB,  
Lenovo,  
Intel Corporation,  
Microsoft Corporation,  
Motorola, Inc.,  
Nokia Corporation,  
Toshiba Corporation

\*Third-party brands and names are the property of their respective owners.

# MASTER TABLE OF CONTENTS

*This table of contents (TOC) covers the entire Bluetooth Specification. In addition each volume has a TOC and each part of a volume is preceded by a detailed TOC.*





## THE BLUETOOTH SPECIFICATION MASTER TABLE OF CONTENTS

---

In the following Master Table of Contents:

- The TOC for each Volume starts at the top of a page.
- The Volume number in black is followed by the Volume name in red.

**Note:** Each Volume is a self contained book which is published and updated separately and is equipped with a TOC of its own. However, this Master TOC is also revised as soon as any of the other Volumes are updated.

- A Volume contains one or more Parts (A, B, etc.); each Part can be viewed independently and has its own TOC.

Red or blue text on the following pages indicates hypertext links that take you directly to the indicated section, on condition that you have access to a complete specification.






---

**Specification Volume 0**
**Master Table of Contents & Compliance Requirements**


---

**Part A****MASTER TABLE OF CONTENTS****Part B****BLUETOOTH COMPLIANCE REQUIREMENTS**

|          |                                                             |           |
|----------|-------------------------------------------------------------|-----------|
| <b>1</b> | <b>Introduction .....</b>                                   | <b>43</b> |
| <b>2</b> | <b>Scope .....</b>                                          | <b>44</b> |
| <b>3</b> | <b>Definitions .....</b>                                    | <b>45</b> |
| 3.1      | Types of Bluetooth Products .....                           | 45        |
| <b>4</b> | <b>Core Configurations.....</b>                             | <b>50</b> |
| 4.1      | Basic Rate Core Configuration.....                          | 50        |
| 4.2      | Enhanced Data Rate Core Configurations.....                 | 51        |
| 4.3      | High Speed Core Configuration .....                         | 52        |
| 4.4      | Low Energy Core Configuration .....                         | 52        |
| 4.5      | Basic Rate and Low Energy Combined Core Configuration ..... | 53        |
| 4.6      | Host Controller Interface Core Configuration .....          | 54        |

**Part C****APPENDIX**

|          |                                                           |           |
|----------|-----------------------------------------------------------|-----------|
| <b>1</b> | <b>Revision History .....</b>                             | <b>59</b> |
| 1.1      | [Vol 0] Master TOC & Compliance Requirements .....        | 59        |
| 1.2      | [Vol 1] Architecture & Terminology Overview .....         | 59        |
| 1.3      | [Vols 2, 3, & 6] Core System Package .....                | 61        |
| 1.4      | [Vol 4] Transport Layers .....                            | 64        |
| <b>2</b> | <b>Contributors.....</b>                                  | <b>65</b> |
| 2.1      | [Vol 0] Master TOC & Compliance Requirements .....        | 65        |
| 2.2      | [Vol 1] Architecture & Terminology Overview .....         | 65        |
| 2.3      | [Vol 2] Core System Package, Controller.....              | 69        |
| 2.4      | [Vol 3] Core System Package, Host .....                   | 87        |
| 2.5      | [Vol 4] Host Controller Interface [Transport Layer] ..... | 98        |
| 2.6      | [Vol 5] Core System Package [AMP Controller volume] ..... | 100       |
| 2.7      | [Vol 6] Low Energy Specification .....                    | 101       |




---

**Specification Volume 1**
**Architecture & Terminology Overview**


---

**Part A**
**ARCHITECTURE**

|          |                                                   |           |
|----------|---------------------------------------------------|-----------|
| <b>1</b> | <b>General Description .....</b>                  | <b>17</b> |
| 1.1      | Overview of BR/EDR Operation .....                | 18        |
| 1.2      | Overview Of Bluetooth Low Energy Operation .....  | 20        |
| 1.3      | Overview Of AMP Operation .....                   | 22        |
| 1.4      | Nomenclature .....                                | 24        |
| <b>2</b> | <b>Core System Architecture .....</b>             | <b>30</b> |
| 2.1      | Core Architectural Blocks .....                   | 34        |
| <b>3</b> | <b>Data Transport Architecture .....</b>          | <b>39</b> |
| 3.1      | Core Traffic Bearers .....                        | 40        |
| 3.2      | Transport Architecture Entities .....             | 45        |
| 3.3      | Physical Channels .....                           | 49        |
| 3.4      | Physical Links .....                              | 58        |
| 3.5      | Logical Links and Logical Transports .....        | 61        |
| 3.6      | L2CAP Channels .....                              | 72        |
| <b>4</b> | <b>Communication Topology and Operation .....</b> | <b>73</b> |
| 4.1      | Piconet Topology .....                            | 73        |
| 4.2      | Operational Procedures and Modes .....            | 76        |
| <b>5</b> | <b>Security Overview .....</b>                    | <b>85</b> |
| 5.1      | BR/EDR Secure Simple Pairing .....                | 85        |
| 5.2      | LE Security .....                                 | 89        |
| 5.3      | AMP Security .....                                | 91        |
| <b>6</b> | <b>Bluetooth Application Architecture .....</b>   | <b>93</b> |
| 6.1      | Bluetooth Profiles .....                          | 93        |
| 6.2      | Generic Access Profile .....                      | 93        |
| 6.3      | Profile Hierarchy .....                           | 94        |
| 6.4      | Generic Attribute Profile .....                   | 95        |
| 6.5      | GATT-based Profile Hierarchy .....                | 96        |

**Part B**
**ACRONYMS & ABBREVIATIONS**

|          |                                                 |            |
|----------|-------------------------------------------------|------------|
| <b>1</b> | <b>List of Acronyms and Abbreviations .....</b> | <b>101</b> |
|----------|-------------------------------------------------|------------|

**Part C**
**CORE SPECIFICATION CHANGE HISTORY**



|          |                                                   |            |
|----------|---------------------------------------------------|------------|
| <b>1</b> | <b>Deprecated Features.....</b>                   | <b>116</b> |
| <b>2</b> | <b>Changes from V1.1 to V1.2.....</b>             | <b>117</b> |
| 2.1      | New Features.....                                 | 117        |
| 2.2      | Structure Changes .....                           | 117        |
| 2.3      | Deprecated Features list .....                    | 117        |
| 2.4      | Changes in Wording.....                           | 118        |
| 2.5      | Nomenclature Changes .....                        | 118        |
| <b>3</b> | <b>Changes from V1.2 to V2.0 + EDR .....</b>      | <b>119</b> |
| 3.1      | New Features.....                                 | 119        |
| 3.2      | Deprecated Features .....                         | 119        |
| <b>4</b> | <b>Changes from V2.0 + EDR to V2.1 + EDR.....</b> | <b>120</b> |
| 4.1      | New features .....                                | 120        |
| 4.2      | Deprecated Features .....                         | 120        |
| <b>5</b> | <b>Changes From V2.1 + EDR To V3.0 + HS .....</b> | <b>121</b> |
| 5.1      | New Features.....                                 | 121        |
| 5.2      | Deprecated Features .....                         | 121        |
| <b>6</b> | <b>Changes From V3.0 + HS To v4.0 .....</b>       | <b>122</b> |
| 6.1      | New Features.....                                 | 122        |
| 6.2      | Deprecated Features .....                         | 122        |

## Part D

### MIXING OF SPECIFICATION VERSIONS

|          |                                               |            |
|----------|-----------------------------------------------|------------|
| <b>1</b> | <b>Mixing of Specification Versions .....</b> | <b>126</b> |
| 1.1      | Features and their Types .....                | 127        |
| 1.2      | Core Specification Addendums .....            | 128        |

## Part E

### IEEE LANGUAGE

|          |                                   |            |
|----------|-----------------------------------|------------|
| <b>1</b> | <b>Use of IEEE Language .....</b> | <b>135</b> |
| 1.1      | Shall .....                       | 135        |
| 1.2      | Must .....                        | 136        |
| 1.3      | Will .....                        | 136        |
| 1.4      | Should .....                      | 136        |
| 1.5      | May .....                         | 136        |
| 1.6      | Can .....                         | 137        |




---

**Specification Volume 2**  
**Core System Package**  
**[BR/EDR Controller volume]**

---

**Part A**

**RADIO SPECIFICATION**

|          |                                                     |           |
|----------|-----------------------------------------------------|-----------|
| <b>1</b> | <b>Scope .....</b>                                  | <b>33</b> |
| <b>2</b> | <b>Frequency Bands and Channel Arrangement.....</b> | <b>35</b> |
| <b>3</b> | <b>Transmitter Characteristics .....</b>            | <b>36</b> |
| 3.1      | Basic Rate .....                                    | 37        |
| 3.2      | Enhanced Data Rate .....                            | 39        |
| <b>4</b> | <b>Receiver Characteristics .....</b>               | <b>46</b> |
| 4.1      | Basic Rate .....                                    | 46        |
| 4.2      | Enhanced Data Rate .....                            | 48        |
| <b>5</b> | <b>Appendix A .....</b>                             | <b>51</b> |
| 5.1      | Nominal Test Conditions .....                       | 51        |
| 5.2      | Extreme Test Conditions .....                       | 52        |
| <b>6</b> | <b>Appendix B .....</b>                             | <b>53</b> |
| <b>7</b> | <b>Appendix C .....</b>                             | <b>54</b> |
| 7.1      | Enhanced Data Rate Modulation Accuracy .....        | 54        |

**Part B**

**BASEBAND SPECIFICATION**

|          |                                           |           |
|----------|-------------------------------------------|-----------|
| <b>1</b> | <b>General Description .....</b>          | <b>65</b> |
| 1.1      | Bluetooth Clock.....                      | 66        |
| 1.2      | Bluetooth Device Addressing.....          | 68        |
| 1.3      | Access Codes .....                        | 69        |
| <b>2</b> | <b>Physical Channels .....</b>            | <b>70</b> |
| 2.1      | Physical Channel Definition .....         | 71        |
| 2.2      | Basic Piconet Physical Channel .....      | 71        |
| 2.3      | Adapted Piconet Physical Channel.....     | 76        |
| 2.4      | Page Scan Physical Channel .....          | 77        |
| 2.5      | Inquiry Scan Physical Channel .....       | 81        |
| 2.6      | Hop Selection .....                       | 83        |
| <b>3</b> | <b>Physical Links .....</b>               | <b>96</b> |
| 3.1      | Link Supervision .....                    | 96        |
| <b>4</b> | <b>Logical Transports .....</b>           | <b>97</b> |
| 4.1      | General .....                             | 97        |
| 4.2      | Logical Transport Address (LT_ADDR) ..... | 97        |



|          |                                                            |            |
|----------|------------------------------------------------------------|------------|
| 4.3      | Synchronous Logical Transports.....                        | 98         |
| 4.4      | Asynchronous Logical Transport.....                        | 98         |
| 4.5      | Transmit/Receive Routines .....                            | 99         |
| 4.6      | Active Slave Broadcast Transport.....                      | 104        |
| 4.7      | Parked Slave Broadcast Transport .....                     | 105        |
| <b>5</b> | <b>Logical Links .....</b>                                 | <b>106</b> |
| 5.1      | Link Control Logical Link (LC).....                        | 106        |
| 5.2      | ACL Control Logical Link (ACL-C) .....                     | 106        |
| 5.3      | User Asynchronous/Isochronous Logical Link (ACL-U) .....   | 106        |
| 5.4      | User Synchronous Data Logical Link (SCO-S) .....           | 107        |
| 5.5      | User Extended Synchronous Data Logical Link (eSCO-S) ..... | 107        |
| 5.6      | Logical Link Priorities .....                              | 107        |
| <b>6</b> | <b>Packets.....</b>                                        | <b>108</b> |
| 6.1      | General Format.....                                        | 108        |
| 6.2      | Bit Ordering .....                                         | 109        |
| 6.3      | Access Code .....                                          | 110        |
| 6.4      | Packet Header .....                                        | 115        |
| 6.5      | Packet Types .....                                         | 117        |
| 6.6      | Payload Format.....                                        | 127        |
| 6.7      | Packet Summary.....                                        | 132        |
| <b>7</b> | <b>Bitstream Processing .....</b>                          | <b>134</b> |
| 7.1      | Error Checking .....                                       | 135        |
| 7.2      | Data Whitening.....                                        | 138        |
| 7.3      | Error Correction.....                                      | 139        |
| 7.4      | FEC Code: Rate 1/3.....                                    | 139        |
| 7.5      | FEC Code: Rate 2/3.....                                    | 140        |
| 7.6      | ARQ Scheme .....                                           | 141        |
| 7.7      | Erroneous Synchronous Data Reporting .....                 | 149        |
| <b>8</b> | <b>Link Controller Operation.....</b>                      | <b>150</b> |
| 8.1      | Overview of States .....                                   | 150        |
| 8.2      | Standby State .....                                        | 151        |
| 8.3      | Connection Establishment Substates .....                   | 151        |
| 8.4      | Device Discovery Substates .....                           | 160        |
| 8.5      | Connection State.....                                      | 165        |
| 8.6      | Active Mode .....                                          | 166        |
| 8.7      | Sniff Mode .....                                           | 180        |
| 8.8      | Hold Mode.....                                             | 183        |
| 8.9      | Park State.....                                            | 184        |
| <b>9</b> | <b>Audio .....</b>                                         | <b>192</b> |
| 9.1      | LOG PCM CODEC.....                                         | 192        |
| 9.2      | CVSD CODEC .....                                           | 192        |



|           |                                 |            |
|-----------|---------------------------------|------------|
| 9.3       | Error Handling.....             | 195        |
| 9.4       | General Audio Requirements..... | 195        |
| <b>10</b> | <b>List of Figures .....</b>    | <b>196</b> |
| <b>11</b> | <b>List of Tables .....</b>     | <b>199</b> |

## Part C

### LINK MANAGER PROTOCOL SPECIFICATION

|          |                                            |            |
|----------|--------------------------------------------|------------|
| <b>1</b> | <b>Introduction .....</b>                  | <b>211</b> |
| <b>2</b> | <b>General Rules .....</b>                 | <b>212</b> |
| 2.1      | Message Transport .....                    | 212        |
| 2.2      | Synchronization .....                      | 212        |
| 2.3      | Packet Format .....                        | 213        |
| 2.4      | Transactions .....                         | 214        |
| 2.5      | Error Handling.....                        | 215        |
| 2.6      | Procedure Rules .....                      | 216        |
| 2.7      | General Response Messages .....            | 217        |
| 2.8      | LMP Message Constraints.....               | 217        |
| <b>3</b> | <b>Device Features .....</b>               | <b>218</b> |
| 3.1      | General Description .....                  | 218        |
| 3.2      | Feature Definitions.....                   | 218        |
| 3.3      | Feature Mask Definition.....               | 225        |
| 3.4      | Link Manager Interoperability policy ..... | 227        |
| <b>4</b> | <b>Procedure Rules .....</b>               | <b>228</b> |
| 4.1      | Connection Control .....                   | 228        |
| 4.2      | Security .....                             | 247        |
| 4.3      | Informational Requests .....               | 275        |
| 4.4      | Role Switch .....                          | 281        |
| 4.5      | Modes of Operation .....                   | 284        |
| 4.6      | Logical Transports .....                   | 297        |
| 4.7      | Test Mode .....                            | 305        |
| <b>5</b> | <b>Summary.....</b>                        | <b>310</b> |
| 5.1      | PDU Summary .....                          | 310        |
| 5.2      | Parameter Definitions .....                | 320        |
| 5.3      | LMP Encapsulated.....                      | 331        |
| 5.4      | Default Values.....                        | 332        |
| <b>6</b> | <b>List of Figures .....</b>               | <b>333</b> |
| <b>7</b> | <b>List of Tables .....</b>                | <b>337</b> |

## Part D

### ERROR CODES



|          |                                                                          |            |
|----------|--------------------------------------------------------------------------|------------|
| <b>1</b> | <b>Overview of Error Codes .....</b>                                     | <b>343</b> |
| 1.1      | Usage Descriptions .....                                                 | 343        |
| 1.2      | HCI Command Errors.....                                                  | 343        |
| 1.3      | List of Error Codes .....                                                | 344        |
| <b>2</b> | <b>Error Code Descriptions.....</b>                                      | <b>347</b> |
| 2.1      | Unknown HCI Command (0X01).....                                          | 347        |
| 2.2      | Unknown Connection Identifier (0X02) .....                               | 347        |
| 2.3      | Hardware Failure (0X03).....                                             | 347        |
| 2.4      | Page Timeout (0X04) .....                                                | 347        |
| 2.5      | Authentication Failure (0X05).....                                       | 347        |
| 2.6      | PIN or key Missing (0X06) .....                                          | 347        |
| 2.7      | Memory Capacity Exceeded (0X07) .....                                    | 347        |
| 2.8      | Connection Timeout (0X08) .....                                          | 348        |
| 2.9      | Connection Limit Exceeded (0X09).....                                    | 348        |
| 2.10     | Synchronous Connection Limit to a Device Exceeded (0X0A) .....           | 348        |
| 2.11     | ACL Connection Already Exists (0X0B) .....                               | 348        |
| 2.12     | Command Disallowed (0X0C).....                                           | 348        |
| 2.13     | Connection Rejected due to Limited Resources (0X0D).....                 | 348        |
| 2.14     | Connection Rejected due to Security Reasons (0X0E) .....                 | 348        |
| 2.15     | Connection Rejected due to Unacceptable BD_ADDR (0X0F).....              | 349        |
| 2.16     | Connection Accept Timeout Exceeded (0X10) .....                          | 349        |
| 2.17     | Unsupported Feature or Parameter Value (0X11).....                       | 349        |
| 2.18     | Invalid HCI Command Parameters (0X12).....                               | 349        |
| 2.19     | Remote User Terminated Connection (0X13) .....                           | 349        |
| 2.20     | Remote Device Terminated Connection due to Low Resources<br>(0X14) ..... | 350        |
| 2.21     | Remote Device Terminated Connection due to Power Off (0X15).<br>.....    | 350        |
| 2.22     | Connection Terminated by Local Host (0X16).....                          | 350        |
| 2.23     | Repeated Attempts (0X17).....                                            | 350        |
| 2.24     | Pairing not Allowed (0X18).....                                          | 350        |
| 2.25     | Unknown LMP PDU (0X19) .....                                             | 350        |
| 2.26     | Unsupported Remote Feature / Unsupported LMP Feature<br>(0X1A).....      | 350        |
| 2.27     | SCO Offset Rejected (0X1B) .....                                         | 350        |
| 2.28     | SCO Interval Rejected (0X1C).....                                        | 351        |
| 2.29     | SCO Air Mode Rejected (0X1D) .....                                       | 351        |
| 2.30     | Invalid LMP Parameters (0X1E).....                                       | 351        |
| 2.31     | Unspecified Error (0X1F) .....                                           | 351        |
| 2.32     | Unsupported LMP Parameter Value (0X20).....                              | 351        |
| 2.33     | Role Change Not Allowed (0X21) .....                                     | 351        |
| 2.34     | LMP Response Timeout / LL Response Timeout (0X22).....                   | 351        |
| 2.35     | LMP Error Transaction Collision (0X23).....                              | 352        |



|      |                                                                      |     |
|------|----------------------------------------------------------------------|-----|
| 2.36 | LMP PDU Not Allowed (0X24).....                                      | 352 |
| 2.37 | Encryption Mode Not Acceptable (0X25).....                           | 352 |
| 2.38 | Link Key cannot be Changed (0X26).....                               | 352 |
| 2.39 | Requested QoS Not Supported (0X27) .....                             | 352 |
| 2.40 | Instant Passed (0X28) .....                                          | 352 |
| 2.41 | Pairing with Unit Key Not Supported (0X29).....                      | 352 |
| 2.42 | Different Transaction Collision (0x2A) .....                         | 352 |
| 2.43 | QoS Unacceptable Parameter (0X2C).....                               | 352 |
| 2.44 | QoS Rejected (0X2D).....                                             | 353 |
| 2.45 | Channel Assessment Not Supported (0X2E) .....                        | 353 |
| 2.46 | Insufficient Security (0X2F).....                                    | 353 |
| 2.47 | Parameter out of Mandatory Range (0X30).....                         | 353 |
| 2.48 | Role Switch Pending (0X32).....                                      | 353 |
| 2.49 | Reserved Slot Violation (0X34).....                                  | 353 |
| 2.50 | Role Switch Failed (0X35) .....                                      | 353 |
| 2.51 | Extended Inquiry Response Too Large (0x36) .....                     | 353 |
| 2.52 | Simple Pairing Not Supported By Host (0X37) .....                    | 354 |
| 2.53 | Host Busy–Pairing(0X38) .....                                        | 354 |
| 2.54 | Connection Rejected Due To No Suitable Channel Found (0X39)<br>..... | 354 |
| 2.55 | Controller Busy (0X3A).....                                          | 354 |
| 2.56 | Unacceptable Connection Interval (0X3B).....                         | 354 |
| 2.57 | Directed Advertising Timeout (0X3C) .....                            | 354 |
| 2.58 | Connection Terminated Due To MIC Failure (0X3D) .....                | 354 |
| 2.59 | Connection Failed To Be Established (0X3E).....                      | 355 |
| 2.60 | MAC Connection Failed (0x3F) .....                                   | 355 |

## Part E

### HOST CONTROLLER INTERFACE FUNCTIONAL SPECIFICATION

|          |                                                          |            |
|----------|----------------------------------------------------------|------------|
| <b>1</b> | <b>Introduction .....</b>                                | <b>369</b> |
| 1.1      | Lower Layers of the Bluetooth Software Stack .....       | 370        |
| <b>2</b> | <b>Overview of Host Controller Transport Layer .....</b> | <b>372</b> |
| 2.1      | Host Controller Transport Layer and AMPS .....           | 372        |
| <b>3</b> | <b>Overview of Commands and Events .....</b>             | <b>373</b> |
| 3.1      | Generic Events .....                                     | 374        |
| 3.2      | Device Setup .....                                       | 374        |
| 3.3      | Controller Flow Control .....                            | 375        |
| 3.4      | Controller Information .....                             | 376        |
| 3.5      | Controller Configuration .....                           | 377        |
| 3.6      | Device Discovery .....                                   | 380        |
| 3.7      | Connection Setup .....                                   | 382        |
| 3.8      | Remote Information .....                                 | 387        |



|          |                                                |            |
|----------|------------------------------------------------|------------|
| 3.9      | Synchronous Connections .....                  | 388        |
| 3.10     | Connection State .....                         | 389        |
| 3.11     | Piconet Structure .....                        | 392        |
| 3.12     | Quality of Service .....                       | 393        |
| 3.13     | Physical Links .....                           | 395        |
| 3.14     | Host Flow Control .....                        | 397        |
| 3.15     | Link Information .....                         | 398        |
| 3.16     | Authentication and Encryption .....            | 400        |
| 3.17     | Testing .....                                  | 405        |
| 3.18     | Alphabetical List of Commands and Events ..... | 408        |
| 3.19     | LE Controller Requirements .....               | 415        |
| <b>4</b> | <b>HCI Flow Control .....</b>                  | <b>418</b> |
| 4.1      | Host to Controller Data Flow Control .....     | 418        |
| 4.2      | Controller to Host Data Flow Control .....     | 421        |
| 4.3      | Disconnection Behavior .....                   | 421        |
| 4.4      | Command Flow Control .....                     | 422        |
| 4.5      | Command Error Handling .....                   | 422        |
| <b>5</b> | <b>HCI Data Formats .....</b>                  | <b>424</b> |
| 5.1      | Introduction .....                             | 424        |
| 5.2      | Data and Parameter Formats .....               | 424        |
| 5.3      | Handles .....                                  | 425        |
| 5.4      | Exchange of HCI-Specific Information .....     | 427        |
| <b>6</b> | <b>HCI Configuration Parameters .....</b>      | <b>435</b> |
| 6.1      | Scan Enable .....                              | 435        |
| 6.2      | Inquiry Scan Interval .....                    | 435        |
| 6.3      | Inquiry Scan Window .....                      | 436        |
| 6.4      | Inquiry Scan Type .....                        | 436        |
| 6.5      | Inquiry Mode .....                             | 436        |
| 6.6      | Page Timeout .....                             | 437        |
| 6.7      | Connection Accept Timeout .....                | 437        |
| 6.8      | Page Scan Interval .....                       | 438        |
| 6.9      | Page Scan Window .....                         | 438        |
| 6.10     | Page Scan Period Mode (Deprecated) .....       | 438        |
| 6.11     | Page Scan Type .....                           | 439        |
| 6.12     | Voice Setting .....                            | 439        |
| 6.13     | PIN Type .....                                 | 440        |
| 6.14     | Link Key .....                                 | 440        |
| 6.15     | Failed Contact Counter .....                   | 440        |
| 6.16     | Authentication Enable .....                    | 441        |
| 6.17     | Hold Mode Activity .....                       | 441        |
| 6.18     | Link Policy Settings .....                     | 443        |
| 6.19     | Flush Timeout .....                            | 444        |



|           |                                                                                   |            |
|-----------|-----------------------------------------------------------------------------------|------------|
| 6.20      | Num Broadcast Retransmissions .....                                               | 444        |
| 6.21      | Link Supervision Timeout.....                                                     | 445        |
| 6.22      | Synchronous Flow Control Enable .....                                             | 445        |
| 6.23      | Local Name.....                                                                   | 446        |
| 6.24      | Extended Inquiry Response.....                                                    | 446        |
| 6.25      | Erroneous Data Reporting .....                                                    | 446        |
| 6.26      | Class Of Device .....                                                             | 447        |
| 6.27      | Supported Commands.....                                                           | 447        |
| 6.28      | Logical Link Accept Timeout .....                                                 | 454        |
| 6.29      | Location Domain Aware.....                                                        | 455        |
| 6.30      | Location Domain .....                                                             | 455        |
| 6.31      | Location Domain Options .....                                                     | 455        |
| 6.32      | Location Options.....                                                             | 456        |
| 6.33      | Flow Control Mode.....                                                            | 456        |
| 6.34      | LE Supported Host .....                                                           | 457        |
| 6.35      | Simultaneous LE Host .....                                                        | 457        |
| <b>7</b>  | <b>HCI Commands and Events .....</b>                                              | <b>458</b> |
| 7.1       | Link Control Commands .....                                                       | 458        |
| 7.2       | Link Policy Commands .....                                                        | 535        |
| 7.3       | Controller & Baseband Commands .....                                              | 560        |
| 7.4       | Informational Parameters .....                                                    | 670        |
| 7.5       | Status Parameters .....                                                           | 680        |
| 7.6       | Testing Commands .....                                                            | 704        |
| 7.7       | Events.....                                                                       | 715        |
| 7.8       | LE Controller Commands .....                                                      | 806        |
| <b>8</b>  | <b>List of Figures .....</b>                                                      | <b>857</b> |
| <b>9</b>  | <b>List of Tables .....</b>                                                       | <b>858</b> |
| <b>10</b> | <b>Appendix A: Deprecated Commands, Events and Configuration Parameters .....</b> | <b>859</b> |
| 10.1      | Read Page Scan Mode Command .....                                                 | 860        |
| 10.2      | Write Page Scan Mode Command .....                                                | 861        |
| 10.3      | Read Page Scan Period Mode Command.....                                           | 862        |
| 10.4      | Write Page Scan Period Mode Command .....                                         | 863        |
| 10.5      | Add SCO Connection Command.....                                                   | 864        |
| 10.6      | Page Scan Mode Change Event .....                                                 | 866        |
| 10.7      | Read Country Code Command .....                                                   | 867        |
| 10.8      | Read Encryption Mode Command.....                                                 | 868        |
| 10.9      | Write Encryption Mode Command .....                                               | 868        |
| 10.10     | Deprecated Parameters.....                                                        | 869        |

## Part F

### MESSAGE SEQUENCE CHARTS



|          |                                                                          |            |
|----------|--------------------------------------------------------------------------|------------|
| <b>1</b> | <b>Introduction .....</b>                                                | <b>877</b> |
| 1.1      | Notation .....                                                           | 877        |
| 1.2      | Flow of Control .....                                                    | 878        |
| 1.3      | Example MSC .....                                                        | 878        |
| <b>2</b> | <b>Services Without Connection Request .....</b>                         | <b>879</b> |
| 2.1      | Remote Name Request .....                                                | 879        |
| 2.2      | One-time Inquiry .....                                                   | 881        |
| 2.3      | Periodic Inquiry .....                                                   | 883        |
| <b>3</b> | <b>ACL Connection Establishment and Detachment .....</b>                 | <b>885</b> |
| 3.1      | Connection Setup .....                                                   | 886        |
| <b>4</b> | <b>Optional Activities After ACL Connection Establishment .....</b>      | <b>893</b> |
| 4.1      | Authentication Requested .....                                           | 893        |
| 4.2      | Simple Pairing Message Sequence Charts .....                             | 894        |
| 4.3      | Link Supervision Timeout Changed Event .....                             | 907        |
| 4.4      | Set Connection Encryption .....                                          | 908        |
| 4.5      | Change Connection Link Key .....                                         | 909        |
| 4.6      | Change Connection Link Key with Encryption Pause and Resume .....        | 909        |
| 4.7      | Master Link Key .....                                                    | 911        |
| 4.8      | Read Remote Supported Features .....                                     | 913        |
| 4.9      | Read Remote Extended Features .....                                      | 913        |
| 4.10     | Read Clock Offset .....                                                  | 914        |
| 4.11     | Role Switch on an Encrypted Link using Encryption Pause and Resume ..... | 914        |
| 4.12     | Refreshing Encryption Keys .....                                         | 915        |
| 4.13     | Read Remote Version Information .....                                    | 916        |
| 4.14     | QOS Setup .....                                                          | 916        |
| 4.15     | Switch Role .....                                                        | 917        |
| 4.16     | AMP Physical Link Creation and Disconnect .....                          | 918        |
| 4.17     | AMP Test Mode Sequence Charts .....                                      | 925        |
| <b>5</b> | <b>Synchronous Connection Establishment and Detachment .....</b>         | <b>929</b> |
| 5.1      | Synchronous Connection Setup .....                                       | 929        |
| <b>6</b> | <b>Sniff, Hold and Park .....</b>                                        | <b>934</b> |
| 6.1      | Sniff Mode .....                                                         | 934        |
| 6.2      | Hold Mode .....                                                          | 935        |
| 6.3      | Park State .....                                                         | 937        |
| <b>7</b> | <b>Buffer Management, Flow Control .....</b>                             | <b>940</b> |
| <b>8</b> | <b>Loopback Mode .....</b>                                               | <b>942</b> |
| 8.1      | Local Loopback Mode .....                                                | 942        |
| 8.2      | Remote Loopback Mode .....                                               | 944        |
| <b>9</b> | <b>List of Figures .....</b>                                             | <b>946</b> |

**Part G****SAMPLE DATA**

|           |                                                |             |
|-----------|------------------------------------------------|-------------|
| <b>1</b>  | <b>Encryption Sample Data .....</b>            | <b>953</b>  |
| 1.1       | Generating Kc' from Kc, .....                  | 953         |
| 1.2       | First Set of Sample Data .....                 | 956         |
| 1.3       | Second Set of Sample Data .....                | 964         |
| 1.4       | Third Set of Samples .....                     | 972         |
| 1.5       | Fourth Set of Samples .....                    | 980         |
| <b>2</b>  | <b>Frequency Hopping Sample Data .....</b>     | <b>988</b>  |
| 2.1       | First set .....                                | 989         |
| 2.2       | Second set .....                               | 995         |
| 2.3       | Third set .....                                | 1001        |
| <b>3</b>  | <b>Access Code Sample Data .....</b>           | <b>1007</b> |
| <b>4</b>  | <b>HEC and Packet Header Sample Data .....</b> | <b>1010</b> |
| <b>5</b>  | <b>CRC Sample Data .....</b>                   | <b>1011</b> |
| <b>6</b>  | <b>Complete Sample Packets .....</b>           | <b>1012</b> |
| 6.1       | Example of DH1 Packet .....                    | 1012        |
| 6.2       | Example of DM1 Packet .....                    | 1013        |
| <b>7</b>  | <b>Simple Pairing Sample Data .....</b>        | <b>1014</b> |
| 7.1       | P-192 Sample Data .....                        | 1014        |
| 7.2       | Hash Functions Sample Data .....               | 1015        |
| <b>8</b>  | <b>Whitening Sequence Sample Data .....</b>    | <b>1026</b> |
| <b>9</b>  | <b>FEC Sample Data .....</b>                   | <b>1029</b> |
| <b>10</b> | <b>Encryption Key Sample Data .....</b>        | <b>1030</b> |
| 10.1      | Four Tests of E1 .....                         | 1030        |
| 10.2      | Four Tests of E21 .....                        | 1035        |
| 10.3      | Three Tests of E22 .....                       | 1037        |
| 10.4      | Tests of E22 With Pin Augmenting .....         | 1039        |
| 10.5      | Four Tests of E3 .....                         | 1049        |

**Part H****SECURITY SPECIFICATION**

|          |                                               |             |
|----------|-----------------------------------------------|-------------|
| <b>1</b> | <b>Security Overview .....</b>                | <b>1059</b> |
| 1.1      | Pausing Encryption and Role Switch .....      | 1060        |
| 1.2      | Change Connection Link Keys .....             | 1060        |
| 1.3      | Periodically Refreshing Encryption Keys ..... | 1060        |
| <b>2</b> | <b>Random Number Generation .....</b>         | <b>1061</b> |
| <b>3</b> | <b>Key Management .....</b>                   | <b>1063</b> |
| 3.1      | Key Types .....                               | 1063        |



|          |                                                             |             |
|----------|-------------------------------------------------------------|-------------|
| 3.2      | Key Generation and Initialization .....                     | 1065        |
| <b>4</b> | <b>Encryption.....</b>                                      | <b>1072</b> |
| 4.1      | Encryption Key Size Negotiation.....                        | 1073        |
| 4.2      | Encryption of Broadcast Messages.....                       | 1073        |
| 4.3      | Encryption Concept.....                                     | 1074        |
| 4.4      | Encryption Algorithm .....                                  | 1075        |
| 4.5      | LFSR Initialization .....                                   | 1078        |
| 4.6      | Key Stream Sequence .....                                   | 1081        |
| <b>5</b> | <b>Authentication .....</b>                                 | <b>1082</b> |
| 5.1      | Repeated Attempts .....                                     | 1084        |
| <b>6</b> | <b>The Authentication And Key-Generating Functions.....</b> | <b>1085</b> |
| 6.1      | The Authentication Function E1 .....                        | 1085        |
| 6.2      | The Functions Ar and A'r.....                               | 1087        |
| 6.3      | E2-Key Generation Function for Authentication.....          | 1089        |
| 6.4      | E3-Key Generation Function for Encryption.....              | 1091        |
| <b>7</b> | <b>Secure Simple Pairing .....</b>                          | <b>1092</b> |
| 7.1      | Phase 1: Public Key Exchange .....                          | 1093        |
| 7.2      | Phase 2: Authentication Stage 1 .....                       | 1094        |
| 7.3      | Phase 3: Authentication Stage 2 .....                       | 1100        |
| 7.4      | Phase 4: Link Key Calculation .....                         | 1101        |
| 7.5      | Phase 5: LMP Authentication and Encryption.....             | 1101        |
| 7.6      | Elliptic Curve Definition .....                             | 1101        |
| 7.7      | Cryptographic Function Definitions .....                    | 1102        |
| <b>8</b> | <b>AMP Security .....</b>                                   | <b>1108</b> |
| 8.1      | Creation of the Initial Generic AMP Link Key .....          | 1108        |
| 8.2      | Creation of Dedicated AMP Link Keys .....                   | 1108        |
| 8.3      | Debug Considerations.....                                   | 1109        |
| <b>9</b> | <b>List of Figures.....</b>                                 | <b>1111</b> |




---

**Specification Volume 3**
**Core System Package**
**[Host volume]**


---

**Part A**
**LOGICAL LINK CONTROL AND ADAPTATION PROTOCOL  
SPECIFICATION**

|          |                                                                                      |           |
|----------|--------------------------------------------------------------------------------------|-----------|
| <b>1</b> | <b>Introduction .....</b>                                                            | <b>30</b> |
| 1.1      | L2CAP Features .....                                                                 | 30        |
| 1.2      | Assumptions .....                                                                    | 33        |
| 1.3      | Scope.....                                                                           | 34        |
| 1.4      | Terminology .....                                                                    | 34        |
| <b>2</b> | <b>General Operation.....</b>                                                        | <b>38</b> |
| 2.1      | Channel Identifiers .....                                                            | 38        |
| 2.2      | Operation Between Devices .....                                                      | 38        |
| 2.3      | Operation Between Layers .....                                                       | 40        |
| 2.4      | Modes of Operation .....                                                             | 41        |
| 2.5      | Mapping Channels to Logical Links .....                                              | 43        |
| <b>3</b> | <b>Data Packet Format.....</b>                                                       | <b>44</b> |
| 3.1      | Connection-oriented Channels in Basic L2CAP Mode .....                               | 44        |
| 3.2      | Connectionless Data Channel in Basic L2CAP Mode .....                                | 45        |
| 3.3      | Connection-oriented Channel in Retransmission/Flow Control/<br>Streaming Modes ..... | 45        |
| <b>4</b> | <b>Signaling Packet Formats .....</b>                                                | <b>54</b> |
| 4.1      | Command Reject (code 0x01) .....                                                     | 56        |
| 4.2      | Connection Request (code 0x02) .....                                                 | 58        |
| 4.3      | Connection Response (code 0x03) .....                                                | 59        |
| 4.4      | Configuration Request (code 0x04) .....                                              | 60        |
| 4.5      | Configuration Response (code 0x05) .....                                             | 62        |
| 4.6      | Disconnection Request (code 0x06).....                                               | 64        |
| 4.7      | Disconnection Response (code 0x07) .....                                             | 65        |
| 4.8      | Echo Request (code 0x08) .....                                                       | 66        |
| 4.9      | Echo Response (code 0x09) .....                                                      | 66        |
| 4.10     | Information Request (code 0x0A) .....                                                | 66        |
| 4.11     | Information Response (code 0x0B) .....                                               | 67        |
| 4.12     | Extended Feature Mask.....                                                           | 69        |
| 4.13     | Fixed Channels Supported .....                                                       | 70        |
| 4.14     | Create Channel Request (code 0x0C).....                                              | 70        |
| 4.15     | Create Channel Response (code 0x0D).....                                             | 71        |
| 4.16     | Move Channel Request (code 0x0E).....                                                | 73        |
| 4.17     | Move Channel Response (code 0x0F) .....                                              | 74        |



|           |                                                                                  |            |
|-----------|----------------------------------------------------------------------------------|------------|
| 4.18      | Move Channel Confirmation (code 0x10).....                                       | 75         |
| 4.19      | Move Channel Confirmation Response (code 0x11).....                              | 76         |
| 4.20      | Connection Parameter Update Request (code 0x12).....                             | 76         |
| 4.21      | Connection Parameter Update Response (code 0x13).....                            | 78         |
| <b>5</b>  | <b>Configuration Parameter Options.....</b>                                      | <b>79</b>  |
| 5.1       | Maximum Transmission Unit (MTU).....                                             | 79         |
| 5.2       | Flush Timeout Option.....                                                        | 81         |
| 5.3       | Quality of Service (QoS) Option.....                                             | 82         |
| 5.4       | Retransmission and Flow Control Option.....                                      | 86         |
| 5.5       | Frame Check Sequence (FCS) Option.....                                           | 91         |
| 5.6       | Extended Flow Specification Option.....                                          | 92         |
| 5.7       | Extended Window Size Option.....                                                 | 96         |
| <b>6</b>  | <b>State Machine.....</b>                                                        | <b>98</b>  |
| 6.1       | General rules for the state machine:.....                                        | 98         |
| 6.2       | Timers events.....                                                               | 114        |
| <b>7</b>  | <b>General Procedures.....</b>                                                   | <b>120</b> |
| 7.1       | Configuration Process.....                                                       | 120        |
| 7.2       | Fragmentation and Recombination.....                                             | 126        |
| 7.3       | Encapsulation of SDUs.....                                                       | 128        |
| 7.4       | Delivery of Erroneous L2CAP SDUs.....                                            | 132        |
| 7.5       | Operation with Flushing On ACL-U Logical Links.....                              | 132        |
| 7.6       | Connectionless Data Channel.....                                                 | 133        |
| 7.7       | Operation Collision Resolution.....                                              | 135        |
| 7.8       | Aggregating Best Effort Extended Flow Specifications.....                        | 135        |
| 7.9       | Prioritizing Data over HCI.....                                                  | 136        |
| 7.10      | Supporting Extended Flow Specification for BR/EDR And BR/EDR/LE Controllers..... | 137        |
| <b>8</b>  | <b>Procedures for Flow Control and Retransmission.....</b>                       | <b>139</b> |
| 8.1       | Information Retrieval.....                                                       | 139        |
| 8.2       | Function of PDU Types for Flow Control and Retransmission.....                   | 139        |
| 8.3       | Variables and Sequence Numbers.....                                              | 140        |
| 8.4       | Retransmission Mode.....                                                         | 144        |
| 8.5       | Flow Control Mode.....                                                           | 150        |
| 8.6       | Enhanced Retransmission Mode.....                                                | 154        |
| 8.7       | Streaming Mode.....                                                              | 184        |
| <b>9</b>  | <b>Procedure for AMP Channel Creation and Handling.....</b>                      | <b>186</b> |
| 9.1       | Create Channel.....                                                              | 186        |
| 9.2       | Move Channel.....                                                                | 189        |
| 9.3       | Disconnect Channel.....                                                          | 197        |
| <b>10</b> | <b>List of Figures.....</b>                                                      | <b>198</b> |
| <b>11</b> | <b>List of Tables.....</b>                                                       | <b>200</b> |

**Part B****SERVICE DISCOVERY PROTOCOL (SDP) SPECIFICATION**

|          |                                                                |            |
|----------|----------------------------------------------------------------|------------|
| <b>1</b> | <b>Introduction .....</b>                                      | <b>209</b> |
| 1.1      | General Description .....                                      | 209        |
| 1.2      | Motivation .....                                               | 209        |
| 1.3      | Requirements.....                                              | 209        |
| 1.4      | Non-requirements and Deferred Requirements.....                | 210        |
| 1.5      | Conventions .....                                              | 210        |
| <b>2</b> | <b>Overview .....</b>                                          | <b>211</b> |
| 2.1      | SDP Client-Server Interaction.....                             | 211        |
| 2.2      | Service Record .....                                           | 212        |
| 2.3      | Service Attribute .....                                        | 213        |
| 2.4      | Service Class.....                                             | 215        |
| 2.5      | Searching for Services.....                                    | 216        |
| 2.6      | Browsing for Services .....                                    | 217        |
| <b>3</b> | <b>Data Representation .....</b>                               | <b>220</b> |
| 3.1      | Data Element .....                                             | 220        |
| 3.2      | Data Element Type Descriptor .....                             | 220        |
| 3.3      | Data Element Size Descriptor.....                              | 221        |
| 3.4      | Data Element Examples .....                                    | 222        |
| <b>4</b> | <b>Protocol Description.....</b>                               | <b>223</b> |
| 4.1      | Transfer Byte Order .....                                      | 223        |
| 4.2      | Protocol Data Unit Format .....                                | 223        |
| 4.3      | Partial Responses and Continuation State .....                 | 225        |
| 4.4      | Error Handling.....                                            | 225        |
| 4.5      | ServiceSearch Transaction.....                                 | 227        |
| 4.6      | ServiceAttribute Transaction.....                              | 230        |
| 4.7      | ServiceSearchAttribute Transaction .....                       | 233        |
| <b>5</b> | <b>Service Attribute Definitions.....</b>                      | <b>237</b> |
| 5.1      | Universal Attribute Definitions.....                           | 237        |
| 5.2      | ServiceDiscoveryServer Service Class Attribute Definitions ... | 246        |
| 5.3      | BrowseGroupDescriptor Service Class Attribute Definitions ...  | 248        |
| <b>6</b> | <b>Security .....</b>                                          | <b>250</b> |

**Part C****GENERIC ACCESS PROFILE**

|          |                               |            |
|----------|-------------------------------|------------|
| <b>1</b> | <b>Introduction .....</b>     | <b>274</b> |
| 1.1      | Scope.....                    | 274        |
| 1.2      | Symbols and Conventions ..... | 275        |
| <b>2</b> | <b>Profile Overview .....</b> | <b>277</b> |



|           |                                                                              |            |
|-----------|------------------------------------------------------------------------------|------------|
| 2.1       | Profile Stack .....                                                          | 277        |
| 2.2       | Profile Roles.....                                                           | 277        |
| 2.3       | User Requirements and Scenarios .....                                        | 280        |
| 2.4       | Profile Fundamentals .....                                                   | 281        |
| 2.5       | Conformance .....                                                            | 281        |
| 2.6       | Other Requirements.....                                                      | 281        |
| <b>3</b>  | <b>User Interface Aspects .....</b>                                          | <b>282</b> |
| 3.1       | The User Interface Level.....                                                | 282        |
| 3.2       | Representation of Bluetooth Parameters .....                                 | 282        |
| 3.3       | Pairing .....                                                                | 286        |
| <b>4</b>  | <b>Modes .....</b>                                                           | <b>287</b> |
| 4.1       | Discoverability Modes .....                                                  | 287        |
| 4.2       | Connectability Modes .....                                                   | 291        |
| 4.3       | Bondable Modes .....                                                         | 293        |
| <b>5</b>  | <b>Security Aspects .....</b>                                                | <b>295</b> |
| 5.1       | Authentication .....                                                         | 295        |
| 5.2       | Security Modes .....                                                         | 296        |
| <b>6</b>  | <b>Idle Mode Procedures .....</b>                                            | <b>314</b> |
| 6.1       | General Inquiry .....                                                        | 314        |
| 6.2       | Limited Inquiry.....                                                         | 315        |
| 6.3       | Name Discovery.....                                                          | 317        |
| 6.4       | Device Discovery .....                                                       | 318        |
| 6.5       | Bonding.....                                                                 | 319        |
| <b>7</b>  | <b>Establishment Procedures .....</b>                                        | <b>323</b> |
| 7.1       | Link Establishment .....                                                     | 323        |
| 7.2       | Channel Establishment .....                                                  | 326        |
| 7.3       | Connection Establishment .....                                               | 328        |
| 7.4       | Establishment of Additional Connection.....                                  | 329        |
| <b>8</b>  | <b>Extended Inquiry Response Data Format .....</b>                           | <b>330</b> |
| 8.1       | EIR Data Type Definitions .....                                              | 331        |
| 8.2       | Example Extended Inquiry Response .....                                      | 333        |
| <b>9</b>  | <b>Operational Modes and Procedures For Use On LE Physical Channels.....</b> | <b>335</b> |
| 9.1       | Broadcast Mode and Observation Procedure .....                               | 335        |
| 9.2       | Discovery Modes and Procedures .....                                         | 337        |
| 9.3       | Connection Modes and Procedures.....                                         | 344        |
| 9.4       | Bonding Modes and Procedures.....                                            | 359        |
| <b>10</b> | <b>LE Security Aspects.....</b>                                              | <b>362</b> |
| 10.1      | Requirements .....                                                           | 362        |
| 10.2      | LE Security Modes .....                                                      | 362        |



|           |                                                                                |            |
|-----------|--------------------------------------------------------------------------------|------------|
| 10.3      | Authentication Procedure .....                                                 | 363        |
| 10.4      | Data Signing .....                                                             | 368        |
| 10.5      | Authorization Procedure .....                                                  | 369        |
| 10.6      | Encryption Procedure .....                                                     | 369        |
| 10.7      | Privacy Feature.....                                                           | 369        |
| 10.8      | Random Device Address .....                                                    | 371        |
| <b>11</b> | <b>Advertising and Scan Response Data Format .....</b>                         | <b>375</b> |
| 11.1      | AD Type Definitions .....                                                      | 375        |
| 11.2      | Example Advertising Data.....                                                  | 377        |
| <b>12</b> | <b>GAP Characteristics for Low Energy .....</b>                                | <b>379</b> |
| 12.1      | Device Name Characteristic .....                                               | 379        |
| 12.2      | Appearance Characteristic .....                                                | 380        |
| 12.3      | Peripheral Privacy Flag Characteristic.....                                    | 380        |
| 12.4      | Reconnection Address Characteristic.....                                       | 380        |
| 12.5      | Peripheral Preferred Connection Parameters Characteristic...                   | 381        |
| <b>13</b> | <b>BR/EDR/LE Operation Modes and Procedure .....</b>                           | <b>383</b> |
| 13.1      | Modes .....                                                                    | 383        |
| 13.2      | Idle Mode Procedures.....                                                      | 385        |
| 13.3      | Establishment Procedures .....                                                 | 388        |
| 13.4      | SDP Interoperability Requirements.....                                         | 388        |
| <b>14</b> | <b>BR/EDR/LE Security Aspects.....</b>                                         | <b>390</b> |
| <b>15</b> | <b>Definitions .....</b>                                                       | <b>391</b> |
| 15.1      | General Definitions .....                                                      | 391        |
| 15.2      | Connection-related Definitions.....                                            | 391        |
| 15.3      | Device-related Definitions .....                                               | 392        |
| 15.4      | Procedure-related Definitions .....                                            | 393        |
| 15.5      | Security-related Definitions .....                                             | 393        |
| <b>16</b> | <b>Appendix A (Normative): Timers and Constants .....</b>                      | <b>395</b> |
| <b>17</b> | <b>Appendix B (Informative): Information Flows of Related Procedures .....</b> | <b>398</b> |
| 17.1      | LMP – Authentication.....                                                      | 398        |
| 17.2      | LMP – Pairing .....                                                            | 399        |
| 17.3      | Service Discovery .....                                                        | 400        |
| 17.4      | Generating a Resolvable Private Address .....                                  | 400        |
| 17.5      | Resolving a Resolvable Private Address .....                                   | 400        |
| <b>18</b> | <b>Appendix C (Normative): EIR and AD Formats .....</b>                        | <b>401</b> |
| 18.1      | Flags .....                                                                    | 401        |
| 18.2      | Service.....                                                                   | 402        |
| 18.3      | Local Name.....                                                                | 402        |
| 18.4      | TX Power Level .....                                                           | 402        |
| 18.5      | Simple Pairing Optional OOB Tags.....                                          | 403        |



|           |                                      |            |
|-----------|--------------------------------------|------------|
| 18.6      | Security Manager TK Value .....      | 403        |
| 18.7      | Security Manager OOB Flags .....     | 403        |
| 18.8      | Slave Connection Interval Range..... | 404        |
| 18.9      | Service Solicitation.....            | 404        |
| 18.10     | Service Data.....                    | 404        |
| 18.11     | Manufacturer Specific Data .....     | 405        |
| <b>19</b> | <b>References .....</b>              | <b>406</b> |

## Part D

### TEST SUPPORT

|          |                                           |            |
|----------|-------------------------------------------|------------|
| <b>1</b> | <b>Test Methodology .....</b>             | <b>411</b> |
| 1.1      | BR/EDR Test Scenarios .....               | 411        |
| 1.2      | AMP Test Scenarios.....                   | 421        |
| 1.3      | References .....                          | 429        |
| <b>2</b> | <b>Test Control Interface (TCI) .....</b> | <b>430</b> |
| 2.1      | Introduction .....                        | 430        |
| 2.2      | TCI Configurations .....                  | 431        |
| 2.3      | TCI Configuration and Usage.....          | 434        |

## Part E

### AMP MANAGER PROTOCOL SPECIFICATION

|          |                                                    |            |
|----------|----------------------------------------------------|------------|
| <b>1</b> | <b>Introduction .....</b>                          | <b>442</b> |
| 1.1      | General Description .....                          | 442        |
| <b>2</b> | <b>General Operation .....</b>                     | <b>443</b> |
| 2.1      | Basic Capabilities .....                           | 443        |
| 2.2      | AMP Manager Channel Over L2CAP .....               | 444        |
| 2.3      | Using the AMP Manager Protocol.....                | 445        |
| 2.4      | Controller IDs .....                               | 447        |
| 2.5      | Controller Types .....                             | 447        |
| <b>3</b> | <b>Protocol Description .....</b>                  | <b>448</b> |
| 3.1      | Packet Formats .....                               | 448        |
| 3.2      | AMP Command Reject (Code 0x01).....                | 449        |
| 3.3      | AMP Discover Request (Code 0x02) .....             | 450        |
| 3.4      | AMP Discover Response (Code 0x03) .....            | 451        |
| 3.5      | AMP Change Notify (Code 0x04).....                 | 455        |
| 3.6      | AMP Change Response (Code 0x05).....               | 456        |
| 3.7      | AMP Get Info Request (Code 0x06) .....             | 456        |
| 3.8      | AMP Get Info Response (Code 0x07).....             | 456        |
| 3.9      | AMP Get AMP Assoc Request (Code 0x08).....         | 458        |
| 3.10     | AMP Get AMP Assoc Response (Code 0x09).....        | 459        |
| 3.11     | AMP Create Physical Link Request (Code 0x0A) ..... | 460        |



|      |                                                        |     |
|------|--------------------------------------------------------|-----|
| 3.12 | AMP Create Physical Link Response (Code 0x0B).....     | 461 |
| 3.13 | AMP Disconnect Physical Link Request (Code 0x0C) ..... | 462 |
| 3.14 | AMP Disconnect Physical Link Response (Code 0x0D)..... | 463 |
| 3.15 | Create Physical Link Collision Resolution .....        | 464 |
| 3.16 | Response Timeout.....                                  | 465 |
| 3.17 | Unexpected BR/EDR Physical Link Disconnect .....       | 465 |

## Part F

### ATTRIBUTE PROTOCOL (ATT)

|          |                                         |            |
|----------|-----------------------------------------|------------|
| <b>1</b> | <b>Introduction .....</b>               | <b>473</b> |
| 1.1      | Scope.....                              | 473        |
| 1.2      | Conformance .....                       | 473        |
| <b>2</b> | <b>Protocol Overview .....</b>          | <b>474</b> |
| <b>3</b> | <b>Protocol Requirements.....</b>       | <b>475</b> |
| 3.1      | Introduction .....                      | 475        |
| 3.2      | Basic Concepts .....                    | 475        |
| 3.3      | Attribute PDU .....                     | 480        |
| 3.4      | Attribute Protocol PDUs .....           | 483        |
| <b>4</b> | <b>Security Considerations.....</b>     | <b>516</b> |
| <b>5</b> | <b>Acronyms and Abbreviations .....</b> | <b>517</b> |

## Part G

### GENERIC ATTRIBUTE PROFILE (GATT)

|          |                                                    |            |
|----------|----------------------------------------------------|------------|
| <b>1</b> | <b>Introduction .....</b>                          | <b>524</b> |
| 1.1      | Scope.....                                         | 524        |
| 1.2      | Profile Dependency .....                           | 524        |
| 1.3      | Conformance .....                                  | 524        |
| 1.4      | Bluetooth Specification Release Compatibility..... | 525        |
| 1.5      | Conventions .....                                  | 525        |
| <b>2</b> | <b>Profile Overview .....</b>                      | <b>526</b> |
| 2.1      | Protocol Stack.....                                | 526        |
| 2.2      | Configurations and Roles .....                     | 526        |
| 2.3      | User Requirements and Scenarios .....              | 527        |
| 2.4      | Profile Fundamentals .....                         | 528        |
| 2.5      | Attribute Protocol .....                           | 528        |
| 2.6      | GATT Profile Hierarchy .....                       | 531        |
| 2.7      | Configured Broadcast .....                         | 533        |
| <b>3</b> | <b>Service Interoperability Requirements .....</b> | <b>535</b> |
| 3.1      | Service Definition .....                           | 535        |
| 3.2      | Include Definition .....                           | 536        |



|           |                                                            |            |
|-----------|------------------------------------------------------------|------------|
| 3.3       | Characteristic Definition .....                            | 536        |
| 3.4       | Summary of GATT Profile Attribute Types .....              | 547        |
| <b>4</b>  | <b>GATT Feature Requirements .....</b>                     | <b>548</b> |
| 4.1       | Overview .....                                             | 548        |
| 4.2       | Feature Support and Procedure Mapping .....                | 548        |
| 4.3       | Server Configuration .....                                 | 550        |
| 4.4       | Primary Service Discovery .....                            | 551        |
| 4.5       | Relationship Discovery .....                               | 554        |
| 4.6       | Characteristic Discovery .....                             | 556        |
| 4.7       | Characteristic Descriptor Discovery .....                  | 558        |
| 4.8       | Characteristic Value Read .....                            | 560        |
| 4.9       | Characteristic Value Write .....                           | 563        |
| 4.10      | Characteristic Value Notification .....                    | 569        |
| 4.11      | Characteristic Value Indications .....                     | 570        |
| 4.12      | Characteristic Descriptors .....                           | 571        |
| 4.13      | GATT Procedure Mapping to ATT Protocol Opcodes .....       | 575        |
| 4.14      | Procedure Timeouts .....                                   | 578        |
| <b>5</b>  | <b>L2CAP Interoperability Requirements .....</b>           | <b>579</b> |
| 5.1       | BR/EDR L2CAP Interoperability Requirements .....           | 579        |
| 5.2       | LE L2CAP Interoperability Requirements .....               | 580        |
| <b>6</b>  | <b>GAP Interoperability Requirements .....</b>             | <b>582</b> |
| 6.1       | BR/EDR GAP Interoperability Requirements .....             | 582        |
| 6.2       | LE GAP Interoperability Requirements .....                 | 582        |
| 6.3       | Disconnected Events .....                                  | 582        |
| <b>7</b>  | <b>Defined Generic Attribute Profile Service .....</b>     | <b>584</b> |
| 7.1       | Service Changed .....                                      | 584        |
| <b>8</b>  | <b>Security Considerations .....</b>                       | <b>586</b> |
| 8.1       | Authentication Requirements .....                          | 586        |
| 8.2       | Authorization Requirements .....                           | 587        |
| <b>9</b>  | <b>SDP Interoperability Requirements .....</b>             | <b>588</b> |
| <b>10</b> | <b>References .....</b>                                    | <b>589</b> |
| <b>11</b> | <b>Appendix: Example Attribute Server Attributes .....</b> | <b>590</b> |

## Part H

### SECURITY MANAGER SPECIFICATION

|          |                               |            |
|----------|-------------------------------|------------|
| <b>1</b> | <b>Introduction .....</b>     | <b>598</b> |
| 1.1      | Scope .....                   | 598        |
| 1.2      | Conventions .....             | 598        |
| <b>2</b> | <b>Security Manager .....</b> | <b>599</b> |
| 2.1      | Introduction .....            | 599        |



|          |                                             |            |
|----------|---------------------------------------------|------------|
| 2.2      | Cryptographic Toolbox .....                 | 600        |
| 2.3      | Pairing Methods .....                       | 603        |
| 2.4      | Security in Bluetooth low energy .....      | 612        |
| <b>3</b> | <b>Security Manager Protocol .....</b>      | <b>620</b> |
| 3.1      | Introduction .....                          | 620        |
| 3.2      | Security Manager Channel over L2CAP .....   | 620        |
| 3.3      | Command Format .....                        | 620        |
| 3.4      | SMP Timeout .....                           | 621        |
| 3.5      | Pairing Methods .....                       | 622        |
| 3.6      | Security in Bluetooth low energy .....      | 628        |
| <b>4</b> | <b>References .....</b>                     | <b>635</b> |
| <b>5</b> | <b>Appendices .....</b>                     | <b>638</b> |
| 5.1      | Appendix A – EDIV and Rand Generation ..... | 638        |
| 5.2      | Appendix B – Key Management .....           | 639        |
| 5.3      | Message Sequence Charts .....               | 643        |




---

**Specification Volume 4**  
**Host Controller Interface**  
**[Transport Layer]**

---

**Part A****UART TRANSPORT LAYER**

|          |                             |           |
|----------|-----------------------------|-----------|
| <b>1</b> | <b>General</b> .....        | <b>13</b> |
| <b>2</b> | <b>Protocol</b> .....       | <b>14</b> |
| <b>3</b> | <b>RS232 Settings</b> ..... | <b>15</b> |
| <b>4</b> | <b>Error Recovery</b> ..... | <b>16</b> |

**Part B****USB TRANSPORT LAYER**

|          |                                                                                           |           |
|----------|-------------------------------------------------------------------------------------------|-----------|
| <b>1</b> | <b>Overview</b> .....                                                                     | <b>21</b> |
| <b>2</b> | <b>USB Endpoint Expectations</b> .....                                                    | <b>25</b> |
|          | 2.1 Descriptor Overview .....                                                             | 25        |
|          | 2.2 Control Endpoint Expectations .....                                                   | 31        |
|          | 2.3 Bulk Endpoints Expectations .....                                                     | 32        |
|          | 2.4 Interrupt Endpoint Expectations .....                                                 | 33        |
|          | 2.5 Isochronous Endpoints Expectations .....                                              | 33        |
| <b>3</b> | <b>Class Code</b> .....                                                                   | <b>34</b> |
|          | 3.1 Bluetooth Codes .....                                                                 | 34        |
| <b>4</b> | <b>Device Firmware Upgrade</b> .....                                                      | <b>36</b> |
| <b>5</b> | <b>Limitations</b> .....                                                                  | <b>37</b> |
|          | 5.1 Power Specific Limitations .....                                                      | 37        |
|          | 5.2 Other Limitations .....                                                               | 37        |
| <b>6</b> | <b>Bluetooth Composite Device Implementation</b> .....                                    | <b>38</b> |
|          | 6.1 Configurations .....                                                                  | 38        |
|          | 6.2 Using USB Interface Association Descriptors for a Primary<br>Controller Function..... | 38        |
|          | 6.3 Combined Primary Controller Function and Single AMP<br>Controller Function.....       | 39        |
| <b>7</b> | <b>References</b> .....                                                                   | <b>43</b> |

**Part C****SECURE DIGITAL (SD)  
TRANSPORT LAYER**

|          |                           |           |
|----------|---------------------------|-----------|
| <b>1</b> | <b>Introduction</b> ..... | <b>48</b> |
|----------|---------------------------|-----------|



|          |                                                      |           |
|----------|------------------------------------------------------|-----------|
| <b>2</b> | <b>Goals .....</b>                                   | <b>49</b> |
| 2.1      | Hardware Goals .....                                 | 49        |
| 2.2      | Software Goals .....                                 | 49        |
| 2.3      | Configuration Goals .....                            | 50        |
| 2.4      | Configuration for Multiple Controllers .....         | 50        |
| <b>3</b> | <b>Physical Interface Documents .....</b>            | <b>51</b> |
| <b>4</b> | <b>Communication .....</b>                           | <b>52</b> |
| 4.1      | Overview .....                                       | 52        |
| <b>5</b> | <b>Appendix A - Acronyms and Abbreviations .....</b> | <b>53</b> |
| <b>6</b> | <b>Appendix B - Related Documents .....</b>          | <b>54</b> |
| <b>7</b> | <b>Appendix C - Tests .....</b>                      | <b>55</b> |
| 7.1      | Test Suite Structure .....                           | 55        |

## Part D

### THREE-WIRE UART TRANSPORT LAYER

|          |                                    |           |
|----------|------------------------------------|-----------|
| <b>1</b> | <b>General .....</b>               | <b>61</b> |
| <b>2</b> | <b>Overview .....</b>              | <b>62</b> |
| <b>3</b> | <b>Slip Layer .....</b>            | <b>63</b> |
| 3.1      | Encoding a Packet .....            | 63        |
| 3.2      | Decoding a Packet .....            | 63        |
| <b>4</b> | <b>Packet Header .....</b>         | <b>65</b> |
| 4.1      | Sequence Number .....              | 65        |
| 4.2      | Acknowledge Number .....           | 66        |
| 4.3      | Data Integrity Check Present ..... | 66        |
| 4.4      | Reliable Packet .....              | 66        |
| 4.5      | Packet Type .....                  | 66        |
| 4.6      | Payload Length .....               | 67        |
| 4.7      | Packet Header Checksum .....       | 67        |
| <b>5</b> | <b>Data Integrity Check .....</b>  | <b>68</b> |
| 5.1      | 16 Bit CCITT-CRC .....             | 68        |
| <b>6</b> | <b>Reliable Packets .....</b>      | <b>69</b> |
| 6.1      | Header Checksum Error .....        | 69        |
| 6.2      | Slip Payload Length Error .....    | 69        |
| 6.3      | Data Integrity Check Error .....   | 69        |
| 6.4      | Out Of Sequence Packet Error ..... | 69        |
| 6.5      | Acknowledgement .....              | 70        |
| 6.6      | Resending Packets .....            | 70        |
| 6.7      | Example Reliable Packet Flow ..... | 70        |
| <b>7</b> | <b>Unreliable Packets .....</b>    | <b>73</b> |



|           |                                     |           |
|-----------|-------------------------------------|-----------|
| 7.1       | Unreliable Packet Header .....      | 73        |
| 7.2       | Unreliable Packet Error .....       | 73        |
| <b>8</b>  | <b>Link Establishment .....</b>     | <b>74</b> |
| 8.1       | Uninitialized State.....            | 75        |
| 8.2       | Initialized State .....             | 75        |
| 8.3       | Active State .....                  | 75        |
| 8.4       | Sync Message .....                  | 76        |
| 8.5       | Sync Response Message .....         | 76        |
| 8.6       | Config Message .....                | 76        |
| 8.7       | Config Response Message .....       | 77        |
| 8.8       | Configuration Field .....           | 77        |
| <b>9</b>  | <b>LOW POWER .....</b>              | <b>80</b> |
| 9.1       | Wakeup Message .....                | 80        |
| 9.2       | Woken Message .....                 | 80        |
| 9.3       | Sleep Message .....                 | 81        |
| <b>10</b> | <b>Out of Frame Control .....</b>   | <b>82</b> |
| 10.1      | Software Flow Control.....          | 82        |
| <b>11</b> | <b>Hardware Configuration .....</b> | <b>83</b> |
| 11.1      | Wires .....                         | 83        |
| 11.2      | Hardware Flow .....                 | 83        |
| <b>12</b> | <b>Recommended Parameters .....</b> | <b>84</b> |
| 12.1      | Timing Parameters .....             | 84        |
| <b>13</b> | <b>References .....</b>             | <b>85</b> |



**Specification Volume 5**  
**Core System Package**  
**[AMP Controller volume]**

**Part A**

**802.11 PROTOCOL ADAPTATION LAYER FUNCTIONAL SPECIFICATION**

|          |                                                         |           |
|----------|---------------------------------------------------------|-----------|
| <b>1</b> | <b>Introduction .....</b>                               | <b>11</b> |
| 1.1      | Organization of the 802.11 PAL .....                    | 11        |
| <b>2</b> | <b>AMP Host Controller Interface .....</b>              | <b>13</b> |
| 2.1      | Read Local Version Information Command .....            | 13        |
| 2.2      | Read Local Amp Info Command .....                       | 13        |
| 2.3      | Reset Command .....                                     | 15        |
| 2.4      | Read Failed Contact Counter Command .....               | 16        |
| 2.5      | Read Link Quality Command .....                         | 16        |
| 2.6      | Read RSSI Command .....                                 | 16        |
| 2.7      | Short Range Mode Command .....                          | 16        |
| 2.8      | Write Best Effort Flush Timeout Command .....           | 17        |
| 2.9      | Read Best Effort Flush Timeout Command .....            | 17        |
| 2.10     | Physical Link Loss Early Warning Event .....            | 17        |
| 2.11     | Physical Link Recovery Event .....                      | 18        |
| 2.12     | Channel Selected Event .....                            | 18        |
| 2.13     | Short Range Mode Change Completed Event .....           | 18        |
| 2.14     | Data Structures .....                                   | 18        |
| 2.15     | Connection Accept Timeout Configuration Parameter ..... | 22        |
| <b>3</b> | <b>Physical Link Manager .....</b>                      | <b>23</b> |
| 3.1      | Physical Link State Machine .....                       | 23        |
| 3.2      | Channel Selection .....                                 | 31        |
| 3.3      | 802.11 Link Creation .....                              | 34        |
| 3.4      | Physical Link Maintenance .....                         | 36        |
| 3.5      | Physical Link Security .....                            | 37        |
| 3.6      | Physical Link Support for QOS .....                     | 39        |
| <b>4</b> | <b>Logical Link Manager .....</b>                       | <b>40</b> |
| 4.1      | Logical Link Creation .....                             | 40        |
| 4.2      | Logical Link Updates .....                              | 41        |
| 4.3      | Logical Link Deletion .....                             | 42        |
| <b>5</b> | <b>Data Manager .....</b>                               | <b>43</b> |
| 5.1      | Encapsulation .....                                     | 43        |
| 5.2      | Coexistence and Local Interference .....                | 44        |
| 5.3      | Explicit Flush .....                                    | 47        |
| 5.4      | Automatic Flush .....                                   | 47        |



|           |                                       |           |
|-----------|---------------------------------------|-----------|
| 5.5       | Quality Of Service Violations.....    | 47        |
| <b>6</b>  | <b>Constants .....</b>                | <b>48</b> |
| <b>7</b>  | <b>Message Sequence Charts .....</b>  | <b>49</b> |
| <b>8</b>  | <b>Appendix A: Test Support .....</b> | <b>50</b> |
| 8.1       | AMP Test Command .....                | 50        |
| 8.2       | AMP Start Test Event .....            | 53        |
| 8.3       | AMP Test End Event .....              | 54        |
| <b>9</b>  | <b>References .....</b>               | <b>55</b> |
| <b>10</b> | <b>List of Figures.....</b>           | <b>56</b> |
| <b>11</b> | <b>List of Tables .....</b>           | <b>57</b> |


**Specification Volume 6**
**Core System Package [Low Energy Controller volume]**
**Part A**
**PHYSICAL LAYER SPECIFICATION**

|          |                                                     |           |
|----------|-----------------------------------------------------|-----------|
| <b>1</b> | <b>Scope .....</b>                                  | <b>14</b> |
| <b>2</b> | <b>Frequency Bands and Channel Arrangement.....</b> | <b>16</b> |
| <b>3</b> | <b>Transmitter Characteristics .....</b>            | <b>17</b> |
| 3.1      | Modulation Characteristics .....                    | 17        |
| 3.2      | Spurious Emissions .....                            | 18        |
| 3.3      | Radio Frequency Tolerance .....                     | 19        |
| <b>4</b> | <b>Receiver Characteristics .....</b>               | <b>20</b> |
| 4.1      | Actual Sensitivity Level .....                      | 20        |
| 4.2      | Interference Performance .....                      | 20        |
| 4.3      | Out-of-Band Blocking.....                           | 21        |
| 4.4      | Intermodulation Characteristics .....               | 21        |
| 4.5      | Maximum Usable Level .....                          | 22        |
| 4.6      | Reference Signal Definition .....                   | 22        |
| <b>5</b> | <b>Appendix A .....</b>                             | <b>23</b> |
| 5.1      | Normal Operating Conditions (NOC) .....             | 23        |
| 5.2      | Extreme Operating Conditions (EOC) .....            | 23        |
| <b>6</b> | <b>Appendix B - Operating Conditions .....</b>      | <b>24</b> |

**Part B**
**LINK LAYER SPECIFICATION**

|          |                                     |           |
|----------|-------------------------------------|-----------|
| <b>1</b> | <b>General Description .....</b>    | <b>30</b> |
| 1.1      | Link Layer States .....             | 30        |
| 1.2      | Bit Ordering.....                   | 32        |
| 1.3      | Device Address.....                 | 33        |
| 1.4      | Physical Channel .....              | 34        |
| <b>2</b> | <b>Air Interface Packets.....</b>   | <b>36</b> |
| 2.1      | Packet Format .....                 | 36        |
| 2.2      | Reserved for Future Use (RFU).....  | 37        |
| 2.3      | Advertising Channel PDU .....       | 37        |
| 2.4      | Data Channel PDU .....              | 44        |
| <b>3</b> | <b>Bit Stream Processing.....</b>   | <b>52</b> |
| 3.1      | Error Checking .....                | 52        |
| 3.2      | Data Whitening .....                | 53        |
| <b>4</b> | <b>Air Interface Protocol .....</b> | <b>55</b> |



|          |                                     |           |
|----------|-------------------------------------|-----------|
| 4.1      | Inter Frame Space .....             | 55        |
| 4.2      | Timing Requirements .....           | 55        |
| 4.3      | Link Layer Device Filtering .....   | 55        |
| 4.4      | Non-Connected States .....          | 57        |
| 4.5      | Connection State .....              | 67        |
| 4.6      | Feature Support .....               | 77        |
| <b>5</b> | <b>Link Layer Control .....</b>     | <b>78</b> |
| 5.1      | Link Layer Control Procedures ..... | 78        |
| 5.2      | Procedure Response Timeout .....    | 87        |

## Part C

### SAMPLE DATA

|          |                                                |           |
|----------|------------------------------------------------|-----------|
| <b>1</b> | <b>Encryption sample data .....</b>            | <b>91</b> |
| 1.1      | Encrypt Command .....                          | 93        |
| 1.2      | Derivation of the MIC and Encrypted Data ..... | 93        |

## Part D

### MESSAGE SEQUENCE CHARTS

|          |                                |            |
|----------|--------------------------------|------------|
| <b>1</b> | <b>Introduction .....</b>      | <b>100</b> |
| 1.1      | Notation .....                 | 100        |
| 1.2      | Control Flow .....             | 101        |
| 1.3      | Example MSC .....              | 101        |
| <b>2</b> | <b>Standby State .....</b>     | <b>102</b> |
| 2.1      | Initial Setup .....            | 102        |
| 2.2      | Random Device Address .....    | 103        |
| 2.3      | White Lists .....              | 103        |
| <b>3</b> | <b>Advertising State .....</b> | <b>104</b> |
| 3.1      | Undirected Advertising .....   | 104        |
| 3.2      | Directed Advertising .....     | 105        |
| <b>4</b> | <b>Scanning State .....</b>    | <b>106</b> |
| 4.1      | Passive Scanning .....         | 106        |
| 4.2      | Active Scanning .....          | 106        |
| <b>5</b> | <b>Initiating State .....</b>  | <b>108</b> |
| 5.1      | Initiating a Connection .....  | 108        |
| 5.2      | Canceling an Initiation .....  | 108        |
| <b>6</b> | <b>Connection State .....</b>  | <b>110</b> |
| 6.1      | Sending Data .....             | 110        |
| 6.2      | Connection Update .....        | 110        |
| 6.3      | Channel Map Update .....       | 111        |
| 6.4      | Features Exchange .....        | 111        |



|      |                                                            |     |
|------|------------------------------------------------------------|-----|
| 6.5  | Version Exchange .....                                     | 112 |
| 6.6  | Start Encryption .....                                     | 113 |
| 6.7  | Start Encryption without Long Term Key .....               | 114 |
| 6.8  | Start Encryption with Event Masked .....                   | 115 |
| 6.9  | Start Encryption Without Slave Supporting Encryption ..... | 115 |
| 6.10 | Restart Encryption .....                                   | 116 |
| 6.11 | Disconnect .....                                           | 116 |

## Part E

### LOW ENERGY LINK LAYER SECURITY

|          |                                                    |            |
|----------|----------------------------------------------------|------------|
| <b>1</b> | <b>Encryption and Authentication Overview.....</b> | <b>121</b> |
| <b>2</b> | <b>CCM.....</b>                                    | <b>122</b> |
| 2.1      | CCM Nonce .....                                    | 122        |
| 2.2      | Counter Mode Blocks .....                          | 123        |
| 2.3      | Encryption Blocks .....                            | 124        |

## Part F

### DIRECT TEST MODE

|          |                                        |            |
|----------|----------------------------------------|------------|
| <b>1</b> | <b>Introduction .....</b>              | <b>128</b> |
| <b>2</b> | <b>Low Energy Test Scenarios .....</b> | <b>129</b> |
| 2.1      | Test Sequences .....                   | 129        |
| 2.2      | Message Sequence Charts.....           | 130        |
| <b>3</b> | <b>UART Test Interface .....</b>       | <b>132</b> |
| 3.1      | UART Interface Characteristics .....   | 132        |
| 3.2      | UART Functional Description.....       | 132        |
| 3.3      | Commands and Events.....               | 133        |
| 3.4      | Events .....                           | 134        |
| 3.5      | Timing – Command and Event .....       | 136        |

## 5 SECURITY OVERVIEW

---

### 5.1 BR/EDR SECURE SIMPLE PAIRING

The primary goal of Secure Simple Pairing is to simplify the pairing procedure for the user. Secondary goals are to maintain or improve the security in Bluetooth wireless technology. Since high levels of security and ease-of-use are often at opposite ends of the spectrum in many technologies and products, much care has been taken to maximize security while minimizing complexity from the end user's point of view.

#### 5.1.1 Security Goals

Secure Simple Pairing has two security goals: protection against passive eavesdropping and protection against man-in-the-middle (MITM) attacks (active eavesdropping). It is a goal of Secure Simple Pairing to exceed the maximum security level provided by the use of a 16 alphanumeric PIN with the pairing algorithm used in Bluetooth Core Specification version 2.0 + EDR and earlier versions. Note that many Bluetooth devices compliant with Bluetooth Core Specification 2.0 + EDR and earlier versions use a 4-digit PIN or a fixed PIN of commonly known values significantly limiting the security on the link.

#### 5.1.2 Passive Eavesdropping Protection

A strong link key coupled with a strong encryption algorithm is necessary to give the user protection against passive eavesdropping. The strength of the link key is based on the amount of entropy (or randomness) in its generation process which would not be known by an attacker. Using legacy pairing, the only source of entropy is the PIN which, in many use cases, is typically four digits either selected by the user or fixed for a given product. Therefore, if the pairing procedure and one authentication exchange is recorded one can run an exhaustive search to find the PIN in a very short amount of time on commonly available computing hardware. With Secure Simple Pairing, the recording attack becomes much harder as the attacker must have solved a hard problem in public key cryptography in order to derive the link key from the recorded information. This protection is independent of the length of the passkey or other numeric values that the user must handle. Secure Simple Pairing gives the same resistance against the recording and passive eavesdropping attacks even when the user is not required to do anything.

Secure Simple Pairing uses Elliptic Curve Diffie Hellman (ECDH) public key cryptography as a means to thwart passive eavesdropping attacks. ECDH provides a very high degree of strength against passive eavesdropping attacks but it may be subject to MITM attacks, which however, are much harder to perform in practice than the passive eavesdropping attack (see [Section 5.1.3, "Man-In-The-Middle Protection," on page 86](#)).



Using the security protocols in the Bluetooth Core Specification version 2.0 + EDR and earlier with a 16 numeric digit PIN achieves about 53 bits of entropy whereas a 16 character alphanumeric, case sensitive PIN yields about 95 bits of entropy when the entire 62 character set is used ([0, ... 9, 'A', ... 'Z', 'a', ... 'z']). Secure Simple Pairing has approximately 95 bits of entropy using the FIPS approved P192 elliptic curve which is at least as good as the entropy in Bluetooth Core Specification 2.0 + EDR and earlier using a 16 character, alphanumeric, case sensitive PIN. Secure Simple Pairing, therefore, exceeds the security requirements of the Bluetooth SIM Access Profile (SAP) which is the profile with the most stringent security requirements. ECDH cryptography was selected over standard Diffie Hellman (often referred to as DH76) since it is computationally less complex and less likely to exceed the low computational capacity in common Bluetooth Controllers.

### 5.1.3 Man-In-The-Middle Protection

A man-in-the-middle (MITM) attack occurs when a user wants to connect two devices but instead of connecting directly with each other they unknowingly connect to a third (attacking) device that plays the role of the device they are attempting to pair with. The third device then relays information between the two devices giving the illusion that they are directly connected. The attacking device may even eavesdrop on communication between the two devices (known as active eavesdropping) and is able to insert and modify information on the connection. In this type of attack, all of the information exchanged between the two devices are compromised and the attacker may inject commands and information into each of the devices thus potentially damaging the function of the devices. Devices falling victim to the attack are capable of communicating only when the attacker is present. If the attacker is not active or out range, the two victim devices will not be able to communicate directly with each other and the user will notice it.

To prevent MITM attacks, Secure Simple Pairing offers two user assisted numeric methods: numerical comparison or passkey entry. If Secure Simple Pairing would use 16 decimal digit numbers, then the usability would be the same as using legacy pairing with 16 decimal digit PIN. The chance for a MITM to succeed inserting its own link keys in this case is a 1 in  $10^{16} = 2^{53}$  pairing instances, which is an unnecessarily low probability.

Secure Simple Pairing protects the user from MITM attacks with a goal of offering a 1 in 1,000,000 chance that a MITM could mount a successful attack. The strength of the MITM protections was selected to minimize the user impact by using a six digit number for numerical comparison and Passkey entry. This level of MITM protection was selected since, in most cases, users can be alerted to the potential presence of a MITM attacker when the connection process fails as a result of a failed MITM attack. While most users feel that provided that they have not compromised their passkey, a 4-digit key is sufficient for authentication (i.e. bank card PIN codes), the use of six digits allows Secure



Simple Pairing to be FIPS compliant and this was deemed to have little perceivable usability impact.

#### **5.1.4 Association Models**

Secure Simple Pairing uses four association models referred to as Numeric Comparison, Just Works, Out Of Band, and Passkey Entry. Each of these association models are described in more detail in the following sections.

The association model used is deterministic based on the I/O capabilities of the two devices.

##### **5.1.4.1 Numeric Comparison**

The Numeric Comparison association model is designed for scenarios where both devices are capable of displaying a six digit number and both are capable of having the user enter "yes" or "no". A good example of this model is the cell phone / PC scenario.

The user is shown a six digit number (from "000000" to "999999") on both displays and then asked whether the numbers are the same on both devices. If "yes" is entered on both devices, the pairing is successful.

The numeric comparison serves two purposes. First, since many devices do not have unique names, it provides confirmation to the user that the correct devices are connected with each other. Second, the numeric comparison provides protection against MITM attacks (see [Section 5.1.3 on page 86](#)).

Note that there is a significant difference from a cryptographic point of view between Numeric Comparison and the PIN entry model used by Bluetooth Core Specification and earlier versions. In the Numeric Comparison association model, the six digit number is an artifact of the security algorithm and not an input to it, as is the case in the Bluetooth security model. Knowing the displayed number is of no benefit in decrypting the encoded data exchanged between the two devices.

##### **5.1.4.2 Just Works**

The Just Works association model is primarily designed for scenarios where at least one of the devices does not have a display capable of displaying a six digit number nor does it have a keyboard capable of entering six decimal digits. A good example of this model is the cell phone/mono headset scenario where most headsets do not have a display.

The Just Works association model uses the Numeric Comparison protocol but the user is never shown a number and the application may simply ask the user to accept the connection (exact implementation is up to the end product manufacturer).



The Just Works association model provides the same protection as the Numeric Comparison association model against passive eavesdropping but offers no protection against the MITM attack.

When compared against today's experience of a headset with a fixed PIN, the security level of the Just Works association model is considerably higher since a high degree of protection against passive eavesdropping is realized.

#### 5.1.4.3 Out of Band

The Out of Band (OOB) association model is primarily designed for scenarios where an Out of Band mechanism is used to both discover the devices as well as to exchange or transfer cryptographic numbers used in the pairing process. In order to be effective from a security point of view, the Out of Band channel should provide different properties in terms of security compared to the Bluetooth radio channel. The Out of Band channel should be resistant to MITM attacks. If it is not, security may be compromised during authentication.

The user's experience differs a bit depending on the Out of Band mechanism. As an example, with a Near Field Communication (NFC) solution, the user(s) will initially touch the two devices together, and is given the option to pair the first device with the other device. If "yes" is entered, the pairing is successful. This is a single touch experience where the exchanged information is used in both devices. The information exchanged includes discovery information (such as the Bluetooth Device Address) as well as cryptographic information. One of the devices will use a Bluetooth Device Address to establish a connection with the other device. The rest of the exchanged information is used during authentication.

The OOB mechanism may be implemented as either read only or read/write. If one side is read only, a one-way authentication is performed. If both sides are read/write, a two-way authentication is performed.

The OOB protocol is selected only when the pairing process has been activated by previous OOB exchange of information and one (or both) of the device(s) gives OOB as the IO capabilities. The protocol uses the information which has been exchanged and simply asks the user to confirm connection.

The OOB association model supports any OOB mechanism where cryptographic information and the Bluetooth Device Address can be exchanged. The OOB association model does not support a solution where the user has activated a Bluetooth connection and would like to use OOB for authentication only.

#### 5.1.4.4 Passkey Entry

The Passkey Entry association model is primarily designed for the scenario where one device has input capability but does not have the capability to dis-



play six digits and the other device has output capabilities. A good example of this model is the PC and keyboard scenario.

The user is shown a six digit number (from "000000" to "999999") on the device with a display, and is then asked to enter the number on the other device. If the value entered on the second device is correct, the pairing is successful. Note that there is a significant difference from a cryptographic point of view between Passkey Entry and the PIN entry model used by Bluetooth Core Specification 2.0 + EDR and earlier versions. In the Passkey Entry association model, the six digit number is independent of the security algorithm and not an input to it, as is the case in the 2.0 + EDR security model. Knowing the entered number is of no benefit in decrypting the encoded data exchanged between the two devices.

#### 5.1.4.5 Association Model Overview

The following diagram shows Secure Simple Pairing from the point of view of the technology used for discovery and then the different association possibilities.

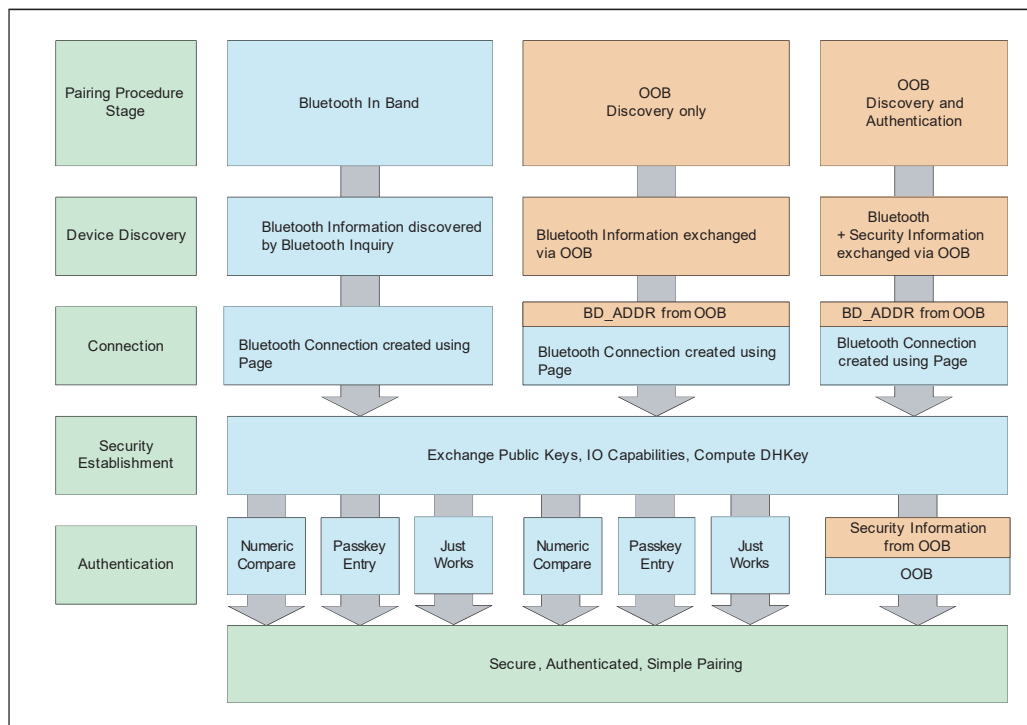


Figure 5.1: Secure Simple Pairing Association Models

## 5.2 LE SECURITY

Bluetooth LE has some differences in security aspects with respect to BR/EDR security features such as Secure Simple Pairing. The association models are



similar to Secure Simple Pairing from the user perspective and have the same names but have differences in the quality of the protection provided.

The overall goal of keeping the cost of the Controller and the complexity of a slave device to a minimum was used in making compromises on security capabilities in LE.

### 5.2.1 Association Models

Bluetooth LE uses three association models referred to as Just Works, Out of Band and Passkey Entry. Bluetooth LE does not have an equivalent of Numeric Comparison. Each of these association models is similar to Secure Simple Pairing with the following exceptions.

- Just Works and Passkey Entry do not provide any passive eavesdropping protection. This is because Secure Simple Pairing uses Elliptic Curve Diffie-Hellman and LE does not.

The use of each association model is based on the I/O capabilities of the devices in a similar manner as Secure Simple Pairing.

### 5.2.2 Key Generation

Key generation in Bluetooth LE is performed by the Host on each LE device independent of any other LE device. Note: Key generation in BR/EDR is performed in the Controller. By performing key generation in the Host, the key generation algorithms can be upgraded without the need to change the Controller.

Bluetooth LE uses multiple keys, each for a specific purpose, as follows:

- Confidentiality of data and device authentication
- Authentication of unencrypted data
- Device Identity

In LE a single link key is generated by combining contributions from each device into a link key used during pairing. In BR/EDR key generation is performed in the Controller.

### 5.2.3 Encryption

Encryption in Bluetooth LE uses AES-CCM cryptography. Like BR/EDR, in LE encryption is performed in the Controller.

### 5.2.4 Signed Data

Bluetooth LE supports the ability to send authenticated data over an unencrypted ATT bearer between two devices with a trusted relationship. This is accomplished by signing the data with a Connection Signature Resolving Key



(CSRK). The sending device places a signature after the Data PDU. The receiving device verifies the signature and if the signature is verified the Data PDU is assumed to come from the trusted source. The signature is composed of a Message Authentication Code generated by the signing algorithm and a counter. The counter is used to protect against a replay attack and is incremented on each signed Data PDU sent.

### 5.2.5 Privacy Feature

Bluetooth LE supports a feature that reduces the ability to track a LE device over a period of time by changing the Bluetooth device address on a frequent basis. The privacy feature is not used in the GAP discovery mode and procedures but it is used when supported during connection mode and connection procedures.

In order for devices using the privacy feature to reconnect to known devices, the device addresses used when the privacy feature is enabled, private address, must be resolvable to the other devices' identity. The private address is generated using the device's identity key exchanged during the bonding procedure. Use of the private address for reconnection is limited to use cases where filtering of devices is disabled. Disabling device filtering may increase the power consumption of the device since the host will need to process all device requests. Situations where the device filtering is disabled should only be used for limited periods of time to minimize the power consumption in this mode.

The privacy feature also defines a reconnection address which allows for a bonded device to reconnect while also filtering devices to known devices. The reconnect address is exchanged between the two devices at each connection. Since reconnect addresses only change between connections, device filtering can be used to minimize processing of excessive requests.

## 5.3 AMP SECURITY

AMP security does not change the user experience because it utilizes the same Secure Simple Pairing association models that were introduced in the Bluetooth 2.1 + EDR Core Specification. From the user's point of view, all radios are "paired" in one process.

AMP security starts during the Secure Simple Pairing process. The link key for BR/EDR is generated during Phase 4 of Secure Simple Pairing. A 256-bit Generic AMP Link Key (GAMP\_LK) is generated from the BR/EDR Link Key. The Generic AMP Link Key is stored in the security database alongside the BR/EDR Link Key once pairing has completed.

AMP security does not affect the BR/EDR Link Key so backwards compatibility is maintained for devices that support the Generic AMP feature with devices that do not.



When an AMP is used for the first time, a dedicated AMP key is created by the AMP Manager for that AMP type using the new Secure Simple Pairing function h2 and a KeyID for the AMP type. The length of the Dedicated AMP Link Key depends on the AMP Type. If the Pair-wise Master key already exists due to a previous connection the AMP link key is not created and the stored key is reused.

The Dedicated AMP Link Key is sent to the PAL during the Physical Link creation process. Each PAL is responsible for using the Dedicated AMP Link Key during the security phase of the physical link establishment process. Note that a Dedicated AMP link key is used for multiple sessions over the same AMP.

Each time a dedicated AMP key is successfully created, the Generic AMP Link Key is updated. This is performed using the h2 function with KeyID "gamp."

# SECURITY SPECIFICATION

*This document describes the specification of the security system which may be used at the link layer. The Encryption, Authentication and Key Generation schemes are specified. The requirements for the supporting process of random number generation are also specified.*



## CONTENTS

|          |                                                             |             |
|----------|-------------------------------------------------------------|-------------|
| <b>1</b> | <b>Security Overview .....</b>                              | <b>1059</b> |
| 1.1      | Pausing Encryption and Role Switch .....                    | 1060        |
| 1.2      | Change Connection Link Keys.....                            | 1060        |
| 1.3      | Periodically Refreshing Encryption Keys .....               | 1060        |
| <b>2</b> | <b>Random Number Generation .....</b>                       | <b>1061</b> |
| <b>3</b> | <b>Key Management.....</b>                                  | <b>1063</b> |
| 3.1      | Key Types .....                                             | 1063        |
| 3.2      | Key Generation and Initialization .....                     | 1065        |
| 3.2.1    | Generation of initialization key, .....                     | 1066        |
| 3.2.2    | Authentication.....                                         | 1066        |
| 3.2.3    | Generation of a unit key .....                              | 1066        |
| 3.2.4    | Generation of a combination key.....                        | 1067        |
| 3.2.5    | Generating the encryption key .....                         | 1068        |
| 3.2.6    | Point-to-multipoint configuration.....                      | 1069        |
| 3.2.7    | Modifying the link keys .....                               | 1069        |
| 3.2.8    | Generating a master key .....                               | 1070        |
| <b>4</b> | <b>Encryption.....</b>                                      | <b>1072</b> |
| 4.1      | Encryption Key Size Negotiation.....                        | 1073        |
| 4.2      | Encryption of Broadcast Messages.....                       | 1073        |
| 4.3      | Encryption Concept.....                                     | 1074        |
| 4.4      | Encryption Algorithm .....                                  | 1075        |
| 4.4.1    | The operation of the cipher .....                           | 1077        |
| 4.5      | LFSR Initialization .....                                   | 1078        |
| 4.6      | Key Stream Sequence .....                                   | 1081        |
| <b>5</b> | <b>Authentication .....</b>                                 | <b>1082</b> |
| 5.1      | Repeated Attempts .....                                     | 1084        |
| <b>6</b> | <b>The Authentication And Key-Generating Functions.....</b> | <b>1085</b> |
| 6.1      | The Authentication Function E1 .....                        | 1085        |
| 6.2      | The Functions Ar and A'r.....                               | 1087        |
| 6.2.1    | The round computations.....                                 | 1087        |
| 6.2.2    | The substitution boxes “e” and “l” .....                    | 1087        |
| 6.2.3    | Key scheduling.....                                         | 1088        |
| 6.3      | E2-Key Generation Function for Authentication.....          | 1089        |
| 6.4      | E3-Key Generation Function for Encryption.....              | 1091        |
| <b>7</b> | <b>Secure Simple Pairing .....</b>                          | <b>1092</b> |
| 7.1      | Phase 1: Public Key Exchange .....                          | 1093        |
| 7.2      | Phase 2: Authentication Stage 1 .....                       | 1094        |



|          |                                                           |             |
|----------|-----------------------------------------------------------|-------------|
| 7.2.1    | Authentication Stage 1: Numeric Comparison Protocol ..... | 1094        |
| 7.2.2    | Authentication Stage 1: Out of Band Protocol.....         | 1096        |
| 7.2.3    | Authentication Stage 1: Passkey Entry Protocol .....      | 1098        |
| 7.3      | Phase 3: Authentication Stage 2.....                      | 1100        |
| 7.4      | Phase 4: Link Key Calculation .....                       | 1101        |
| 7.5      | Phase 5: LMP Authentication and Encryption .....          | 1101        |
| 7.6      | Elliptic Curve Definition .....                           | 1101        |
| 7.7      | Cryptographic Function Definitions .....                  | 1102        |
| 7.7.1    | The Simple Pairing Commitment Function f1.....            | 1102        |
| 7.7.2    | The Simple Pairing Numeric Verification Function g ..     | 1103        |
| 7.7.3    | The Simple Pairing Key Derivation Function f2.....        | 1104        |
| 7.7.4    | The Simple Pairing Check Function f3.....                 | 1105        |
| 7.7.5    | The Simple Pairing AMP Key Derivation Function h2 .....   | 1106        |
| <b>8</b> | <b>AMP Security .....</b>                                 | <b>1108</b> |
| 8.1      | Creation of the Initial Generic AMP Link Key .....        | 1108        |
| 8.2      | Creation of Dedicated AMP Link Keys.....                  | 1108        |
| 8.3      | Debug Considerations .....                                | 1109        |
| <b>9</b> | <b>List of Figures .....</b>                              | <b>1111</b> |

## 1 SECURITY OVERVIEW

Bluetooth wireless technology provides peer-to-peer communications over short distances. In order to provide usage protection and information confidentiality, the system provides security measures both at the application layer and the link layer. These measures are designed to be appropriate for a peer environment. This means that in each device, the authentication and encryption routines are implemented in the same way. Four different entities are used for maintaining security at the link layer: a Bluetooth device address, two secret keys, and a pseudo-random number that shall be regenerated for each new transaction. The four entities and their sizes are summarized in [Table 1.1](#).

| Entity                                                          | Size       |
|-----------------------------------------------------------------|------------|
| BD_ADDR                                                         | 48 bits    |
| Private user key, authentication                                | 128 bits   |
| Private user key, encryption<br>configurable length (byte-wise) | 8-128 bits |
| RAND                                                            | 128 bits   |

*Table 1.1: Entities used in authentication and encryption procedures.*

The Bluetooth device address (BD\_ADDR) is the 48-bit address. The BD\_ADDR can be obtained via user interactions, or, automatically, via an inquiry routine by a device.

The secret keys are derived during initialization and are never disclosed. The encryption key is derived from the authentication key during the authentication process. For the authentication algorithm, the size of the key used is always 128 bits. For the encryption algorithm, the key size may vary between 1 and 16 octets (8 - 128 bits). The size of the encryption key is configurable for two reasons. The first has to do with the many different requirements imposed on cryptographic algorithms in different countries – both with respect to export regulations and official attitudes towards privacy in general. The second reason is to facilitate a future upgrade path for the security without the need of a costly redesign of the algorithms and encryption hardware; increasing the effective key size is the simplest way to combat increased computing power at the opponent side.

The encryption key is entirely different from the authentication key (even though the latter is used when creating the former, as is described in [Section 6.4 on page 1091](#)). Each time encryption is activated, a new encryption key shall be generated. Thus, the lifetime of the encryption key does not necessarily correspond to the lifetime of the authentication key.

It is anticipated that the authentication key will be more static in its nature than the encryption key – once established, the particular application running on the device decides when, or if, to change it. To underline the fundamental impor-



tance of the authentication key to a specific link, it is often referred to as the link key.

The RAND is a pseudo-random number which can be derived from a random or pseudo-random process in the device. This is not a static parameter and will change frequently.

In the remainder of this chapter, the terms user and application are used interchangeably to designate the entity that is at either side.

## 1.1 PAUSING ENCRYPTION AND ROLE SWITCH

To perform a role switch on a connection encryption must be disabled or paused as the encryption is based off the master's clock and Bluetooth address information. Unfortunately, if the role switch is required, and encryption is turned off, the device on the other end of the link will not be aware of the reason for disabling encryption, and can therefore take two possible actions: send clear text data, or disconnect. Neither of these possible actions is desirable. When performing a role switch on an encrypted link, the role switch shall be performed as a single operation where possible. If this is not possible because the other device does not support the encryption pause feature, then when a device wishes to have an encrypted link, and encryption is disabled, then the device should not send any user data, and should not disconnect. If both devices support the encryption pause feature, then this procedure shall be used.

## 1.2 CHANGE CONNECTION LINK KEYS

It is possible to perform a change of connection link keys while a link is encrypted, but it is not possible to use the new link keys until encryption has been stopped and then restarted. As with role switches, disabling encryption and then re-enabling it again can cause user data to be sent in the clear or a disconnection to occur. The use of encryption pausing prevents this problem from occurring. On devices that do not support the encryption pause feature, when a device wishes to have an encrypted link, and encryption is disabled, then the device should not send any data, and should not disconnect. If both devices support the encryption pause feature, then this procedure shall be used.

## 1.3 PERIODICALLY REFRESHING ENCRYPTION KEYS

If both devices support the encryption pause feature, then the encryption keys shall be refreshed by the Link Manager at least once every  $2^{28}$  Bluetooth Clocks (about 23.3 hours) if it is not refreshed by the Host or the remote Link Manager. To refresh an encryption key, the Host maybe use the Change Connection Link Key procedure or request an encryption key refresh. If the encryption key has not been refreshed before going stale, a device may disconnect the link.

## 2 RANDOM NUMBER GENERATION

Each device has a pseudo-random number generator. Pseudo-random numbers are used for many purposes within the security functions – for instance, for the challenge-response scheme, for generating authentication and encryption keys, nonces used in Simple Pairing, for Passkeys used in authentication.

$1/2^L$  A device shall use a pseudo random number generator compliant with [FIPS PUB 140-2] (<http://csrc.nist.gov/publications/fips/fips140-2/fips1402annexc.pdf>)

An example of a possible random generator is provided in [FIPS PUB 186-2] Appendix 3.1 (<http://csrc.nist.gov/publications/fips/archives/fips186-2/fips186-2-change1.pdf>), which can be used by replacing SHA-1 function with the SHA-256 function

The device shall use a seed with at least the minimum entropy required by the pseudo random number generator.

The random number generator shall be tested against the [FIPS SP800-22] (<http://csrc.nist.gov/publications/nistpubs/800-22-rev1/SP800-22rev1.pdf>). This encompasses the verification of the following statistical tests performed on the output of the PRNG as specified by the [FIPS SP800-22]:

1. The Frequency (Monobit) Test
2. Frequency Test within a Block
3. The Runs Test
4. Test for the Longest-Run-of-Ones in a Block
5. The Binary Matrix Rank Test
6. The Discrete Fourier Transform (Spectral) Test
7. The Non-overlapping Template Matching Test
8. The Overlapping Template Matching Test
9. Maurer's "Universal Statistical" Test
10. The Lempel-Ziv Compression Test
11. The Linear Complexity Test
12. The Serial Test
13. The Approximate Entropy Test
14. The Cumulative Sums (Cusums) Test
15. The Random Excursions Test
16. The Random Excursions Variant Test

These tests are part of standard statistical mathematical packages. Some test suites, like the Diehard test suite can be used to verify the compliance. Alternatively, other tools, such as the DieHarder ([http://www.phy.duke.edu/~rgb/General/rand\\_rate.php](http://www.phy.duke.edu/~rgb/General/rand_rate.php)) or the available NIST tools (<http://csrc.nist.gov/groups/ST/>)



[toolkit/random\\_number.html](http://toolkit/random_number.html)) and the corresponding recommendations (<http://csrc.nist.gov/publications/nistpubs/800-22-rev1/SP800-22rev1.pdf>) may also be used.

## 3 KEY MANAGEMENT

---

It is important that the encryption key size within a specific device cannot be set by the user – this should be a factory preset entity. In order to prevent the user from over-riding the permitted key size, the Bluetooth baseband processing shall not accept an encryption key given from higher software layers. Whenever a new encryption key is required, it shall be created as defined in [Section 6.4 on page 1091](#).

Changing a link key shall also be done through the defined baseband procedures. Depending on what kind of link key it is, different approaches are required. The details are found in [Section 3.2.7 on page 1069](#).

### 3.1 KEY TYPES

The link key is a 128-bit random number which is shared between two or more parties and is the base for all security transactions between these parties. The link key itself is used in the authentication routine. Moreover, the link key is used as one of the parameters when the encryption key is derived.

In the following, a session is defined as the time interval for which the device is a member of a particular piconet. Thus, the session terminates when the device disconnects from the piconet.

The link keys are either semi-permanent or temporary. A semi-permanent link key may be stored in non-volatile memory and may be used after the current session is terminated. Consequently, once a semi-permanent link key is defined, it may be used in the authentication of several subsequent connections between the devices sharing it. The designation semi-permanent is justified by the possibility of changing it. How to do this is described in [Section 3.2.7 on page 1069](#).

The lifetime of a temporary link key is limited by the lifetime of the current session – it shall not be reused in a later session. Typically, in a point-to-multipoint configuration where the same information is to be distributed securely to several recipients, a common encryption key is useful. To achieve this, a special link key (denoted master key) may temporarily replace the current link keys. The details of this procedure are found in [Section 3.2.6 on page 1069](#).

In the following, the current link key is the link key in use at the current moment. It can be semi-permanent or temporary. Thus, the current link key is used for all authentications and all generation of encryption keys in the on-going connection (session).



In order to accommodate different types of applications, four types of link keys have been defined:

- the combination key  $K_{AB}$
- the unit key  $K_A$
- the temporary key  $K_{master}$
- the initialization key  $K_{init}$

Note: the use of unit keys is deprecated since it is implicitly insecure.

In addition to these keys there is an encryption key, denoted  $K_C$ . This key is derived from the current link key. Whenever encryption is activated by an LM command, the encryption key shall be changed automatically. The purpose of separating the authentication key and encryption key is to facilitate the use of a shorter encryption key without weakening the strength of the authentication procedure. There are no governmental restrictions on the strength of authentication algorithms. However, in some countries, such restrictions exist on the strength of encryption algorithms.

The combination key  $K_{AB}$  and the unit key  $K_A$  are functionally indistinguishable; the difference is in the way they are generated. The unit key  $K_A$  is generated in, and therefore dependent on, a single device A. The unit key shall be generated once at installation of the device; thereafter, it is very rarely changed. The combination key is derived from information in both devices A and B, and is therefore always dependent on two devices. The combination key is derived for each new combination of two devices.

It depends on the application or the device whether a unit key or a combination key is used. Devices which have little memory to store keys, or are installed in equipment that will be accessible to a large group of users, should use their own unit key. In that case, they only have to store a single key. Applications that require a higher security level should use the combination keys. These applications will require more memory since a combination key for each link to a different device has to be stored.

The master key,  $K_{master}$ , shall only be used during the current session. It shall only replace the original link key temporarily. For example, this may be utilized when a master wants to reach more than two devices simultaneously using the same encryption key, see [Section 3.2.6 on page 1069](#).

The initialization key,  $K_{init}$ , shall be used as the link key during the initialization process when no combination or unit keys have been defined and exchanged yet or when a link key has been lost. The initialization key protects the transfer of initialization parameters. The key is derived from a random number, an L-octet PIN code, and a BD\_ADDR. This key shall only be used during initialization.



The PIN may be a fixed number provided with the device (for example when there is no user interface as in a PSTN plug). Alternatively, the PIN can be selected by the user, and then entered in both devices that are to be matched. The latter procedure should be used when both devices have a user interface, for example a phone and a laptop. Entering a PIN in both devices is more secure than using a fixed PIN in one of the devices, and should be used whenever possible. Even if a fixed PIN is used, it shall be possible to change the PIN; this is in order to prevent re-initialization by users who once obtained the PIN. If no PIN is available, a default value of zero may be used. The length of this default PIN is one byte, PIN (default) = 0x00. This default PIN may be provided by the host.

For many applications the PIN code will be a relatively short string of numbers. Typically, it may consist of only four decimal digits. Even though this gives sufficient security in many cases, there exist countless other, more sensitive, situations where this is not reliable enough. Therefore, the PIN code may be chosen to be any length from 1 to 16 octets. For the longer lengths, the devices exchanging PIN codes may not use mechanical (i.e. human) interaction, but rather may use software at the application layer. For example, this can be a Diffie-Hellman key agreement, where the exchanged key is passed on to the  $K_{init}$  generation process in both devices, just as in the case of a shorter PIN code.

### 3.2 KEY GENERATION AND INITIALIZATION

The link keys must be generated and distributed among the devices in order to be used in the authentication procedure. Since the link keys shall be secret, they shall not be obtainable through an inquiry routine in the same way as the Bluetooth device addresses. The exchange of the keys takes place during an initialization phase which shall be carried out separately for each two devices that are using authentication and encryption. The initialization procedures consist of the following five parts:

- generation of an initialization key
- generation of link key
- link key exchange
- authentication
- generation of encryption key in each device (optional)

After the initialization procedure, the devices can proceed to communicate, or the link can be disconnected. If encryption is implemented, the  $E_0$  algorithm shall be used with the proper encryption key derived from the current link key. For any new connection established between devices A and B, they should use the common link key for authentication, instead of once more deriving  $K_{init}$  from the PIN. A new encryption key derived from that particular link key shall be created next time encryption is activated.

If no link key is available, the LM shall automatically start an initialization procedure.

### 3.2.1 Generation of initialization key, $K_{init}$

A link key is used temporarily during initialization, the initialization key  $K_{init}$ . This key shall be derived by the  $E_{22}$  algorithm from a BD\_ADDR, a PIN code, the length of the PIN (in octets), and a random number IN\_RAND. The principle is depicted in [Figure 6.4 on page 1091](#). The 128-bit output from  $E_{22}$  shall be used for key exchange during the generation of a link key. When the devices have performed the link key exchange, the initialization key shall be discarded.

When the initialization key is generated, the PIN is augmented with the BD\_ADDR. If one device has a fixed PIN the BD\_ADDR of the other device shall be used. If both devices have a variable PIN the BD\_ADDR of the device that received IN\_RAND shall be used. If both devices have a fixed PIN they cannot be paired. Since the maximum length of the PIN used in the algorithm cannot exceed 16 octets, it is possible that not all octets of BD\_ADDR will be used. This procedure ensures that  $K_{init}$  depends on the identity of the device with a variable PIN. A fraudulent device may try to test a large number of PINs by claiming another BD\_ADDR each time. It is the application's responsibility to take countermeasures against this threat. If the device address is kept fixed, the waiting interval before the next try may be increased exponentially (see [Section 5.1 on page 1084](#)).

The details of the  $E_{22}$  algorithm can be found in [Section 6.3 on page 1089](#).

### 3.2.2 Authentication

The authentication procedure shall be carried out as described in [Section 5 on page 1082](#). During each authentication, a new  $AU\_RAND_A$  shall be issued.

Mutual authentication is achieved by first performing the authentication procedure in one direction and then immediately performing the authentication procedure in the opposite direction.

As a side effect of a successful authentication procedure an auxiliary parameter, the Authenticated Ciphering Offset (ACO), will be computed. The ACO shall be used for ciphering key generation as described in [Section 3.2.5 on page 1068](#).

The claimant/verifier status is determined by the LM.

### 3.2.3 Generation of a unit key

A unit key  $K_A$  shall be generated when the device is in operation for the first time; i.e. not during each initialization. The unit key shall be generated by the  $E_{21}$  algorithm as described in [Section 6.3 on page 1089](#). Once created, the unit key should be stored in non-volatile memory and very rarely changed. If after initial-

ization the unit key is changed, any previously initialized devices will possess a wrong link key. At initialization, the application must determine which of the two parties will provide the unit key as the link key. Typically, this will be the device with restricted memory capabilities, since this device only has to remember its own unit key. The unit key shall be transferred to the other party and then stored as the link key for that particular party. So, for example in [Figure 3.1 on page 1067](#), the unit key of device A,  $K_A$ , is being used as the link key for the connection A-B; device A sends the unit key  $K_A$  to device B; device B will store  $K_A$  as the link key  $K_{BA}$ . For another initialization, for example with device C, device A will reuse its unit key  $K_A$ , whereas device C stores it as  $K_{CA}$ .

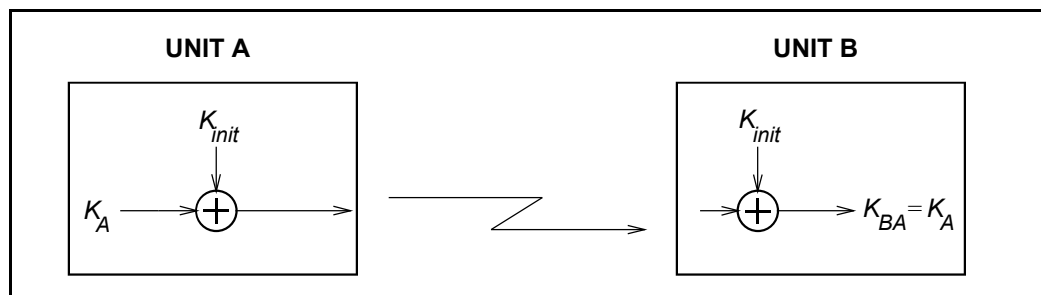


Figure 3.1: Generation of unit key. When the unit key has been exchanged, the initialization key is discarded in both devices.

### 3.2.4 Generation of a combination key

To use a combination key, it is first generated during the initialization procedure. The combination key is the combination of two numbers generated in device A and B, respectively. First, each device shall generate a random number,  $LK\_RAND_A$  and  $LK\_RAND_B$ . Then, utilizing  $E_{2I}$  with the random number and their own  $BD\_ADDRs$ , the two random numbers

$$K\_K_A = E_{2I}(LK\_RAND_A, BD\_ADDR_A) \quad (EQ\ 1)$$

and

$$LK\_K_B = E_{2I}(LK\_RAND_B, BD\_ADDR_B), \quad (EQ\ 2)$$

shall be created in device A and device B, respectively. These numbers constitute the devices' contribution to the combination key that is to be created. Then, the two random numbers  $LK\_RAND_A$  and  $LK\_RAND_B$  shall be exchanged securely by XORing with the current link key,  $K$ . Thus, device A shall send  $K \oplus LK\_RAND_A$  to device B, while device B shall send  $K \oplus LK\_RAND_B$  to device A. If this is done during the initialization phase the link key  $K = K_{init}$ .

When the random numbers  $LK\_RAND_A$  and  $LK\_RAND_B$  have been mutually exchanged, each device shall recalculate the other device's contribution to the combination key. This is possible since each device knows the Bluetooth



device address of the other device. Thus, device A shall calculate (EQ 2) on page 1067 and device B shall calculate (EQ 1) on page 1067. After this, both devices shall combine the two numbers to generate the 128-bit link key. The combining operation is a simple bitwise modulo-2 addition (i.e. XOR). The result shall be stored in device A as the link key  $K_{AB}$  and in device B as the link key  $K_{BA}$ . When both devices have derived the new combination key, a mutual authentication procedure shall be initiated to confirm the success of the transaction. The old link key shall be discarded after a successful exchange of a new combination key. The message flow between master and slave and the principle for creating the combination key is depicted in Figure 3.2 on page 1068.

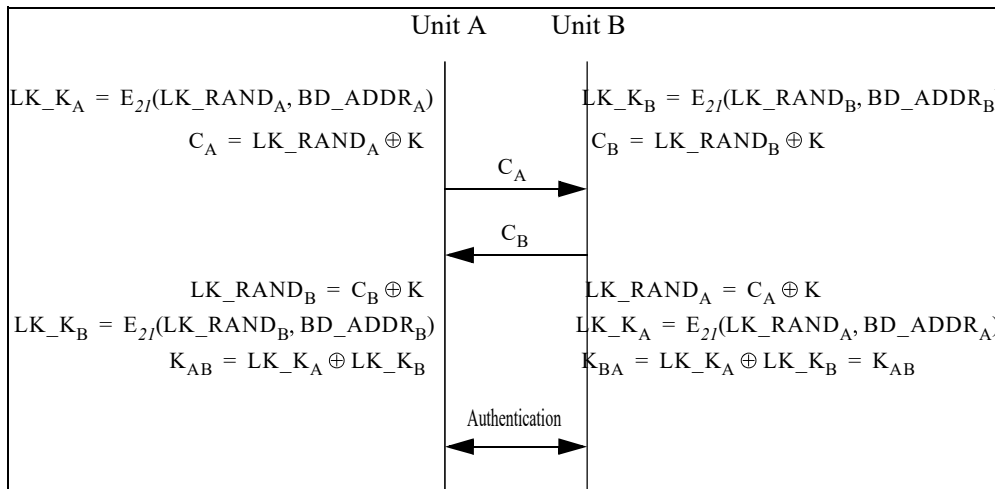


Figure 3.2: Generating a combination key. The old link key ( $K$ ) is discarded after the exchange of a new combination key has succeeded

### 3.2.5 Generating the encryption key

The encryption key,  $K_C$ , is derived by algorithm  $E_3$  from the current link key, a 96-bit Ciphering Offset number (COF), and a 128-bit random number. The COF is determined in one of two ways. If the current link key is a master key, then COF shall be derived from the master BD\_ADDR. Otherwise the value of COF shall be set to the value of ACO as computed during the authentication procedure. Therefore:<sup>1</sup>

$$\text{COF} = \begin{cases} \text{BD\_ADDR} \cup \text{BD\_ADDR}, & \text{if link key is a master key} \\ \text{ACO}, & \text{otherwise.} \end{cases} \quad (\text{EQ } 3)$$

There is an explicit call of  $E_3$  when the LM activates encryption. Consequently, the encryption key is automatically changed each time the device enters

1.  $x \cup y$  denotes the concatenation of  $x$  and  $y$ .



encryption mode. The details of the key generating function  $E_3$  can be found in [Section 6.4 on page 1091](#).

### 3.2.6 Point-to-multipoint configuration

It is possible for the master to use separate encryption keys for each slave in a point-to-multipoint configuration with ciphering activated. Then, if the application requires more than one slave to listen to the same payload, each slave must be addressed individually. This can cause unwanted capacity loss for the piconet. Moreover, a slave might not be capable of switching between two or more encryption keys in real time (e.g., after looking at the LT\_ADDR in the header). Thus, the master cannot use different encryption keys for broadcast messages and individually addressed traffic. Therefore, the master may tell several slave devices to use a common link key (and, hence, indirectly also to use a common encryption key) and may then broadcast the information encrypted. For many applications, this key is only of temporary interest. In the following discussion, this key is denoted by  $K_{\text{master}}$ .

The transfer of necessary parameters shall be protected by the routine described in [Section 3.2.8 on page 1070](#). After the confirmation of successful reception in each slave, the master shall issue a command to the slaves to replace their respective current link key by the new (temporary) master key. Before encryption can be activated, the master shall also generate and distribute a common EN RAND to all participating slaves. Using this random number and the newly derived master key, each slave shall generate a new encryption key.

Note that the master must negotiate the encryption key length to use individually with each slave that will use the master key. If the master has already negotiated with some of these slaves, it has knowledge of the sizes that can be accepted. There may be situations where the permitted key lengths of some devices are incompatible. In that case, the master must exclude the limiting device from the group.

When all slaves have received the necessary data, the master can communicate information on the piconet securely using the encryption key derived from the new temporary link key. Each slave in possession of the master key can eavesdrop on all encrypted traffic, not only the traffic intended for itself. The master may tell all participants to fall back to their old link keys simultaneously.

### 3.2.7 Modifying the link keys

A link key based on a unit key can be changed. The unit key is created once during first use. Typically, the link key should be changed rather than the unit key, as several devices may share the same unit key as link key (e.g. a printer whose unit key is distributed to all users using the printer's unit key as link key). Changing the unit key will require re-initialization of all devices connecting.

Changing the unit key can be justified in some circumstances, e.g. to deny access to all previously allowed devices.

If the key change concerns combination keys, then the procedure is straightforward. The change procedure is identical to the procedure described in [Figure 3.2 on page 1068](#), using the current value of the combination key as link key. This procedure can be carried out at any time after the authentication and encryption start. Since the combination key corresponds to a single link, it can be modified each time this link is established. This will improve the security of the system since then old keys lose their validity after each session.

Starting up an entirely new initialization procedure is also possible. In that case, user interaction is necessary since a PIN will be required in the authentication and encryption procedures.

### 3.2.8 Generating a master key

The key-change routines described so far are semi-permanent. To create the master link key, which can replace the current link key during a session (see [Section 3.2.6 on page 1069](#)), other means are needed. First, the master shall create a new link key from two 128-bit random numbers, RAND1 and RAND2. This shall be done by

$$K_{\text{master}} = E_{22}(\text{RAND1}, \text{RAND2}, 16). \quad (\text{EQ } 4)$$

This key is a 128-bit random number. The reason for using the output of  $E_{22}$  and not directly choosing a random number as the key, is to avoid possible problems with degraded randomness due to a poor implementation of the random number generator within the device.

Then, a third random number, RAND, shall be transmitted to the slave. Using  $E_{22}$  with the current link key and RAND as inputs, both the master and the slave shall compute a 128-bit overlay. The master shall send the bitwise XOR of the overlay and the new link key to the slave. The slave, who knows the overlay, shall recalculate  $K_{\text{master}}$ . To confirm the success of this transaction, the devices shall perform a mutual authentication procedure using the new link key. This procedure shall then be repeated for each slave that receives the new link key. The ACO values from the authentications shall not replace the current ACO, as this ACO is needed to (re)compute a ciphering key when the master falls back to the previous (non-temporary) link key.

The master activates encryption by an LM command. Before activating encryption, the master shall ensure that all slaves receive the same random number, EN\_RAND, since the encryption key is derived through the means of  $E_3$  individually in all participating devices. Each slave shall compute a new encryption key as follows:

$$K_C = E_3(K_{\text{master}}, \text{EN\_RAND}, \text{COF}) \quad (\text{EQ } 5)$$



where the value of COF shall be derived from the master's BD\_ADDR as specified by equation (EQ 3) on page 1068. The details of the encryption key generating function are described in Section 6.4 on page 1091. The message flow between the master and the slave when generating the master key is depicted in Figure 3.3. Note that in this case the ACO produced during the authentication is not used when computing the ciphering key.

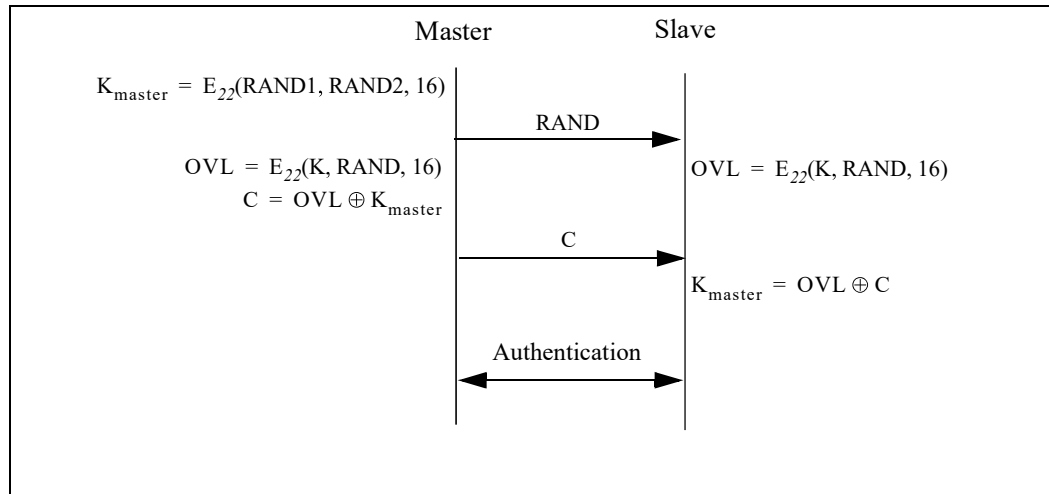


Figure 3.3: Master link key distribution and computation of the corresponding encryption key.

## 4 ENCRYPTION

User information can be protected by encryption of the packet payload; the access code and the packet header shall never be encrypted. The encryption of the payload shall be carried out with a stream cipher called  $E_0$  that shall be re-synchronized for every payload. The overall principle is shown in [Figure 4.1 on page 1072](#).

The stream cipher system  $E_0$  shall consist of three parts:

- the first part performs initialization (generation of the payload key). The payload key generator shall combine the input bits in an appropriate order and shall shift them into the four LFSRs used in the key stream generator.
- the second part generates the key stream bits and shall use a method derived from the summation stream cipher generator attributable to Massey and Rueppel. The second part is the main part of the cipher system, as it will also be used for initialization.
- the third part performs encryption and decryption.

The Massey and Rueppel method has been thoroughly investigated, and there exist good estimates of its strength with respect to presently known methods for cryptanalysis. Although the summation generator has weaknesses that can be used in correlation attacks, the high re-synchronization frequency will disrupt such attacks.

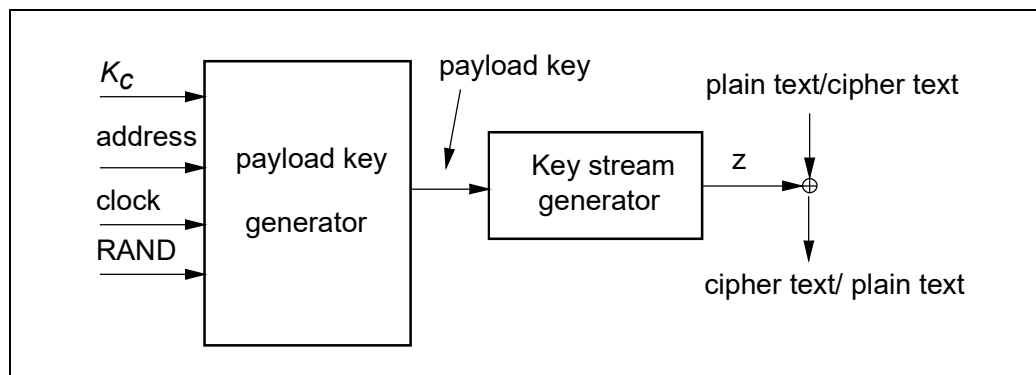


Figure 4.1: Stream ciphering for Bluetooth with  $E_0$ .

## 4.1 ENCRYPTION KEY SIZE NEGOTIATION

Each device implementing the baseband specification shall have a parameter defining the maximal allowed key length,  $L_{\max}$ ,  $1 \leq L_{\max} \leq 16$  (number of octets in the key). For each application using encryption, a number  $L_{\min}$  shall be defined indicating the smallest acceptable key size for that particular application. Before generating the encryption key, the devices involved shall negotiate to decide the key size to use.

The master shall send a suggested value,  $L_{\text{sug}}^{(M)}$ , to the slave. Initially, the suggested value shall be set to  $L_{\max}^{(M)}$ . If  $L_{\min}^{(S)} \leq L_{\text{sug}}^{(M)}$ , and, the slave supports the suggested length, the slave shall acknowledge and this value shall be the length of the encryption key for this link. However, if both conditions are not fulfilled, the slave shall send a new proposal,  $L_{\text{sug}}^{(S)} < L_{\text{sug}}^{(M)}$ , to the master. This value shall be the largest among all supported lengths less than the previous master suggestion. Then, the master shall perform the corresponding test on the slave suggestion. This procedure shall be repeated until a key length agreement is reached, or, one device aborts the negotiation. An abort may be caused by lack of support for  $L_{\text{sug}}$  and all smaller key lengths, or if  $L_{\text{sug}} < L_{\min}$  in one of the devices. In case of an abort link encryption cannot be employed.

The possibility of a failure in setting up a secure link is an unavoidable consequence of letting the application decide whether to accept or reject a suggested key size. However, this is a necessary precaution. Otherwise a fraudulent device could enforce a weak protection on a link by claiming a small maximum key size.

## 4.2 ENCRYPTION OF BROADCAST MESSAGES

There may be three settings for the baseband regarding encryption:

1. No encryption.  
This is the default setting. No messages are encrypted
2. Point-to-point only encryption.  
Broadcast messages are not encrypted. This may be enabled either during the connection establishment procedure or after the connection has been established.
3. Point-to-point and broadcast encryption.  
All messages are encrypted. This may be enabled after the connection has been established only. This setting should not be enabled unless all affected links share the same master link key as well as the same EN\_RANDOM value, both used in generating the encryption key.

### 4.3 ENCRYPTION CONCEPT

| Broadcast traffic               | Individually addressed traffic  |
|---------------------------------|---------------------------------|
| No encryption                   | No encryption                   |
| No encryption                   | Encryption, $K_{\text{master}}$ |
| Encryption, $K_{\text{master}}$ | Encryption, $K_{\text{master}}$ |

Table 4.1: Possible encryption modes for a slave in possession of a master key.

For the encryption routine, a stream cipher algorithm is used in which ciphering bits are bit-wise modulo-2 added to the data stream to be sent over the air interface. The payload is ciphered after the CRC bits are appended, but, prior to the FEC encoding.

Each packet payload shall be ciphered separately. The cipher algorithm  $E_\theta$  uses the master Bluetooth device address (BD\_ADDR), 26 bits of the master real-time clock ( $\text{CLK}_{26-1}$ ) and the encryption key  $K_C$  as input, see [Figure 4.2 on page 1075](#) (where it is assumed that device A is the master).

The encryption key  $K_C$  is derived from the current link key, COF, and a random number,  $\text{EN\_RAND}_A$  (see [Section 6.4 on page 1091](#)). The random number shall be issued by the master before entering encryption mode. Note that  $\text{EN\_RAND}_A$  is publicly known since it is transmitted as plain text over the air.

Within the  $E_\theta$  algorithm, the encryption key  $K_C$  is modified into another key denoted  $K'_C$ . The maximum effective size of this key shall be factory preset and may be set to any multiple of eight between one and sixteen (8-128 bits). The procedure for deriving the key is described in [Section 4.5 on page 1078](#).

The real-time clock is incremented for each slot. The  $E_\theta$  algorithm shall be re-initialized at the start of each new packet (i.e. for Master-to-Slave as well as for Slave-to-Master transmission). By using  $\text{CLK}_{26-1}$  at least one bit is changed between two transmissions. Thus, a new keystream is generated after each re-initialization. For packets covering more than a single slot, the Bluetooth clock as found in the first slot shall be used for the entire packet.

The encryption algorithm  $E_\theta$  generates a binary keystream,  $K_{\text{cipher}}$ , which shall be modulo-2 added to the data to be encrypted. The cipher is symmetric; decryption shall be performed in exactly the same way using the same key as used for encryption.

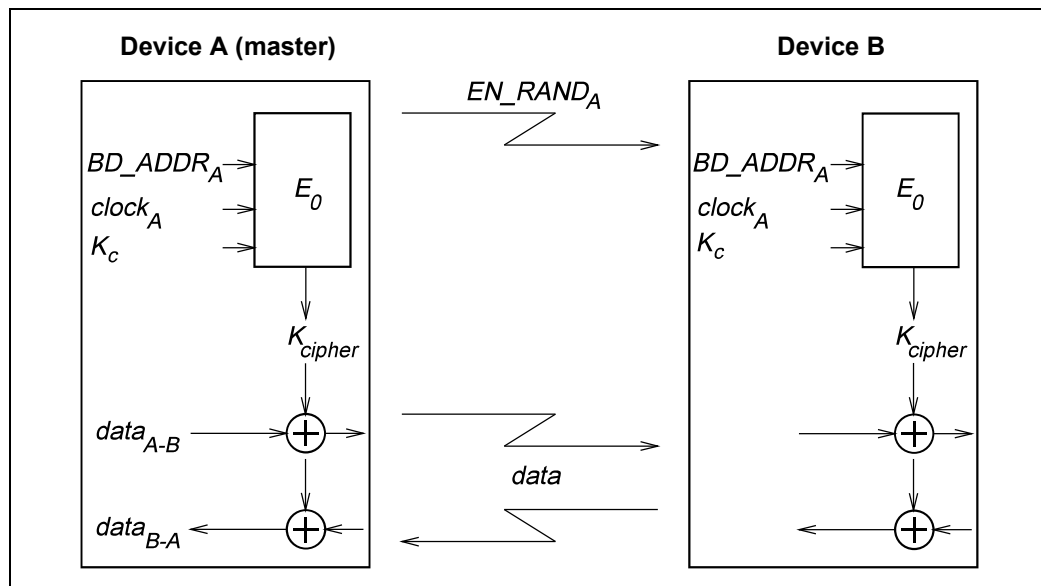


Figure 4.2: Functional description of the encryption procedure

## 4.4 ENCRYPTION ALGORITHM

The system uses linear feedback shift registers (LFSRs) whose output is combined by a simple finite state machine (called the summation combiner) with 16 states. The output of this state machine is the key stream sequence, or, during initialization phase, the randomized initial start value. The algorithm uses an encryption key  $K_C$ , a 48-bit Bluetooth address, the master clock bits  $CLK_{26-1}$ , and a 128-bit RAND value. [Figure 4.3 on page 1076](#) shows the setup.

There are four LFSRs ( $LFSR_1, \dots, LFSR_4$ ) of lengths  $L_1 = 25$ ,  $L_2 = 31$ ,  $L_3 = 33$ , and,  $L_4 = 39$ , with feedback polynomials as specified in [Table 4.2 on page 1076](#). The total length of the registers is 128. These polynomials are all primitive. The Hamming weight of all the feedback polynomials is chosen to be five – a reasonable trade-off between reducing the number of required XOR gates in the hardware implementation and obtaining good statistical properties of the generated sequences.

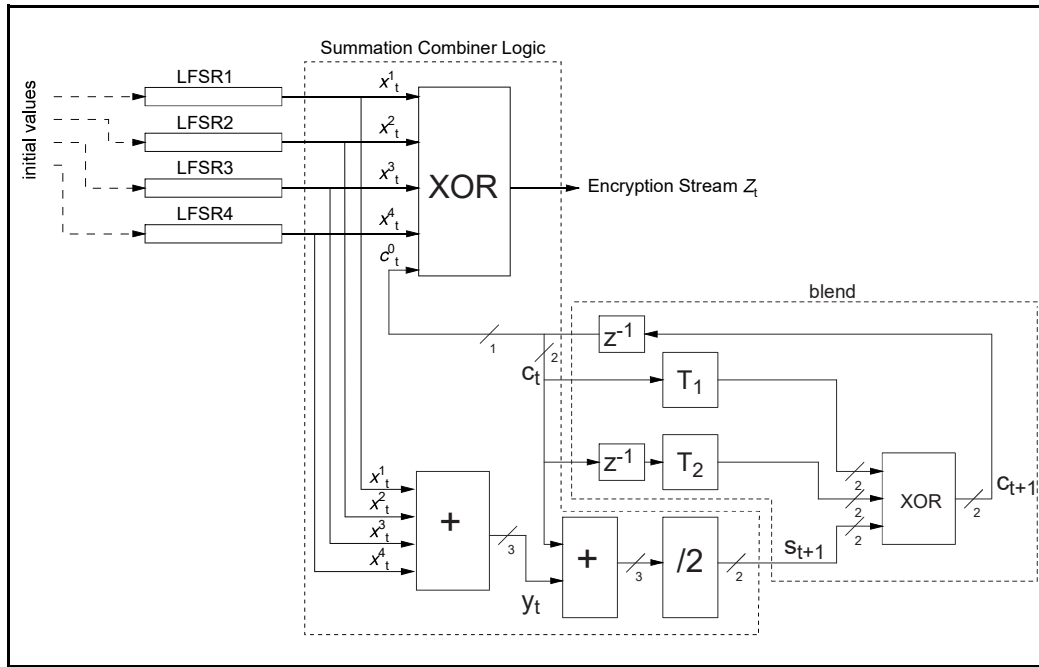


Figure 4.3: Concept of the encryption engine.

| i | $L_i$ | feedback $f_i(t)$                       | weight |
|---|-------|-----------------------------------------|--------|
| 1 | 25    | $t^{25} + t^{20} + t^{12} + t^8 + 1$    | 5      |
| 2 | 31    | $t^{31} + t^{24} + t^{16} + t^{12} + 1$ | 5      |
| 3 | 33    | $t^{33} + t^{28} + t^{24} + t^4 + 1$    | 5      |
| 4 | 39    | $t^{39} + t^{36} + t^{28} + t^4 + 1$    | 5      |

Table 4.2: The four primitive feedback polynomials.

Let  $x_t^i$  denote the  $t^{\text{th}}$  symbol of LFSR <sub>$i$</sub> . The value  $y_t$  is derived from the four-tuple  $x_t^1, \dots, x_t^4$  using the following equation:

$$y_t = \sum_{i=1}^4 x_t^i, \quad (\text{EQ 6})$$

where the sum is over the integers. Thus  $y_t$  can take the values 0,1,2,3, or 4. The output of the summation generator is obtained by the following equations:

$$z_t = x_t^1 \oplus x_t^2 \oplus x_t^3 \oplus x_t^4 \oplus c_t^0 \in \{0, 1\}, \quad (\text{EQ 7})$$



$$s_{t+1} = (s_{t+1}^1, s_{t+1}^0) = \left\lfloor \frac{y_t + c_t}{2} \right\rfloor \in \{0, 1, 2, 3\}, \quad (\text{EQ } 8)$$

$$c_{t+1} = (c_{t+1}^1, c_{t+1}^0) = s_{t+1} \oplus T_1[c_t] \oplus T_2[c_{t-1}], \quad (\text{EQ } 9)$$

where  $T_1[.]$  and  $T_2[.]$  are two different linear bijections over  $\text{GF}(4)$ . Suppose  $\text{GF}(4)$  is generated by the irreducible polynomial  $x^2 + x + 1$ , and let  $\alpha$  be a zero of this polynomial in  $\text{GF}(4)$ . The mappings  $T_1$  and  $T_2$  are now defined as:

$$T_1: \text{GF}(4) \rightarrow \text{GF}(4)$$

$$x \mapsto x$$

$$T_2: \text{GF}(4) \rightarrow \text{GF}(4)$$

$$x \mapsto (\alpha + 1)x.$$

The elements of  $\text{GF}(4)$  can be written as binary vectors. This is summarized in [Table 4.3](#).

| x  | $T_1[x]$ | $T_2[x]$ |
|----|----------|----------|
| 00 | 00       | 00       |
| 01 | 01       | 11       |
| 10 | 10       | 01       |
| 11 | 11       | 10       |

Table 4.3: The mappings  $T_1$  and  $T_2$ .

Since the mappings are linear, they can be implemented using XOR gates; i.e.

$$T_1: (x_1, x_0) \mapsto (x_1, x_0),$$

$$T_2: (x_1, x_0) \mapsto (x_0, x_1 \oplus x_0).$$

#### 4.4.1 The operation of the cipher

[Figure 4.4 on page 1078](#) gives an overview of the operation in time. The encryption algorithm shall run through the initialization phase before the start of transmission or reception of a new packet. Thus, for multislot packets the cipher is initialized using the clock value of the first slot in the multislot sequence.

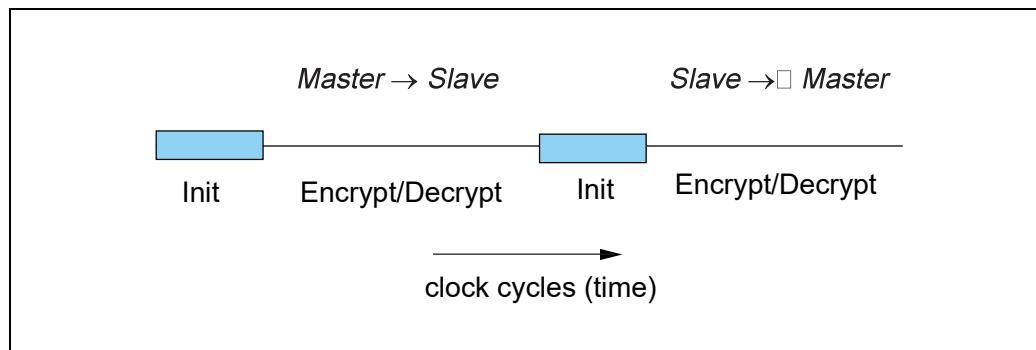


Figure 4.4: Overview of the operation of the encryption engine. Between each start of a packet (TX or RX), the LFSRs are re-initialized.

## 4.5 LFSR INITIALIZATION

The key stream generator is loaded with an initial value for the four LFSRs (in total 128 bits) and the 4 bits that specify the values of  $c_0$  and  $c_{-1}$ . The 132 bit initial value is derived from four inputs by using the key stream generator. The input parameters are the key  $K_C$ , a 128-bit random number RAND, a 48-bit Bluetooth device address, and the 26 master clock bits  $CLK_{26-1}$ .

The effective length of the encryption key may vary between 8 and 128 bits. Note that the actual key length as obtained from  $E_3$  is 128 bits. Then, within  $E_0$ , the key length may be reduced by a modulo operation between  $K_C$  and a polynomial of desired degree. After reduction, the result is encoded with a block code in order to distribute the starting states more uniformly. The operation shall be as defined in (EQ 10) on page 1079.

When the encryption key has been created the LFSRs are loaded with their initial values. Then, 200 stream cipher bits are created by operating the generator. Of these bits, the last 128 are fed back into the key stream generator as an initial value of the four LFSRs. The values of  $c_t$  and  $c_{t-1}$  are kept. From this point on, when clocked the generator produces the encryption (decryption) sequence which is bitwise XORed to the transmitted (received) payload data.

In the following, octet  $i$  of a binary sequence  $X$  is  $X[i]$ . Bit 0 of  $X$  is the LSB. Then, the LSB of  $X[i]$  corresponds to bit  $8i$  of the sequence  $X$ , the MSB of  $X[i]$  is bit  $8i + 7$  of  $X$ . For instance, bit 24 of the Bluetooth device address is the LSB of  $BD\_ADDR[3]$ .

The details of the initialization shall be as follows:

1. Create the encryption key to use from the 128-bit secret key  $K_C$  and the 128-bit publicly known  $EN\_RAND$ . Let  $L$ ,  $1 \leq L \leq 16$ , be the



effective key length in number of octets. The resulting encryption key is  $K'_C$ :

$$K'_C(x) = g_2^{(L)}(x)(K_C(x) \bmod g_1^{(L)}(x)), \quad (\text{EQ 10})$$

where  $\deg(g_1^{(L)}(x)) = 8L$  and  $\deg(g_2^{(L)}(x)) \leq 128 - 8L$ . The polynomials are defined in [Table 4.4](#).

2. Shift the 3 inputs  $K'_C$ , the Bluetooth device address, the clock, and the six-bit constant 111001 into the LFSRs. In total, 208 bits are shifted in:
  - a) Open all switches shown in [Figure 4.5 on page 1080](#);
  - b) Arrange inputs bits as shown in [Figure 4.5](#); Set the content of all shift register elements to zero. Set  $t = 0$ .
  - c) Start shifting bits into the LFSRs. The rightmost bit at each level of [Figure 4.5](#) is the first bit to enter the corresponding LFSR.
  - d) When the first input bit at level  $i$  reaches the rightmost position of  $\text{LFSR}_i$ , close the switch of this LFSR.
  - e) At  $t = 39$  (when the switch of  $\text{LFSR}_4$  is closed), reset both blend registers  $c_{39} = c_{39-1} = 0$ ; Up to this point, the content of  $c_t$  and  $c_{t-1}$  has been of no concern. However, their content will now be used in computing the output sequence.
  - f) From now on output symbols are generated. The remaining input bits are continuously shifted into their corresponding shift registers. When the last bit has been shifted in, the shift register is clocked with input = 0;
 

Note: When finished,  $\text{LFSR}_1$  has effectively clocked 30 times with feedback closed,  $\text{LFSR}_2$  has clocked 24 times,  $\text{LFSR}_3$  has clocked 22 times, and  $\text{LFSR}_4$  has effectively clocked 16 times with feedback closed.
3. To mix initial data, continue to clock until 200 symbols have been produced with all switches closed ( $t = 239$ );
4. Keep blend registers  $c_t$  and  $c_{t-1}$ , make a parallel load of the last 128 generated bits into the LFSRs according to [Figure 4.6](#) at  $t = 240$ ;

After the parallel load in item 4, the blend register contents shall be updated for each subsequent clock.

| L | deg  | $g_1^{(L)}$                         | deg   | $g_2^{(L)}$                         |
|---|------|-------------------------------------|-------|-------------------------------------|
| 1 | [8]  | 00000000 00000000 00000000 0000011d | [119] | 00e275a0 abd218d4 cf928b9b bf6cb08f |
| 2 | [16] | 00000000 00000000 00000000 0001003f | [112] | 0001e3f6 3d7659b3 7f18c258 cff6efef |
| 3 | [24] | 00000000 00000000 00000000 010000db | [104] | 000001be f66c6c3a b1030a5a 1919808b |
| 4 | [32] | 00000000 00000000 00000001 000000af | [96]  | 00000001 6ab89969 de17467f d3736ad9 |

Table 4.4: Polynomials used when creating  $K'_C$ .<sup>1</sup>



| L  | deg   | $g_1^{(L)}$                           | deg  | $g_2^{(L)}$                         |
|----|-------|---------------------------------------|------|-------------------------------------|
| 5  | [40]  | 00000000 00000000 00000100 00000039   | [88] | 00000000 01630632 91da50ec 55715247 |
| 6  | [48]  | 00000000 00000000 00010000 00000291   | [77] | 00000000 00002c93 52aa6cc0 54468311 |
| 7  | [56]  | 00000000 00000000 01000000 00000095   | [71] | 00000000 000000b3 f7ffce2 79f3a073  |
| 8  | [64]  | 00000000 00000001 00000000 0000001b   | [63] | 00000000 00000000 a1ab815b c7ec8025 |
| 9  | [72]  | 00000000 00000100 00000000 00000609   | [49] | 00000000 00000000 0002c980 11d8b04d |
| 10 | [80]  | 00000000 00010000 00000000 00000215   | [42] | 00000000 00000000 0000058e 24f9a4bb |
| 11 | [88]  | 00000000 01000000 00000000 0000013b   | [35] | 00000000 00000000 0000000c a76024d7 |
| 12 | [96]  | 00000001 00000000 00000000 000000dd   | [28] | 00000000 00000000 00000000 1c9c26b9 |
| 13 | [104] | 00000100 00000000 00000000 0000049d   | [21] | 00000000 00000000 00000000 0026d9e3 |
| 14 | [112] | 00010000 00000000 00000000 0000014f   | [14] | 00000000 00000000 00000000 00004377 |
| 15 | [120] | 01000000 00000000 00000000 000000e7   | [7]  | 00000000 00000000 00000000 00000089 |
| 16 | [128] | 1 00000000 00000000 00000000 00000000 | [0]  | 00000000 00000000 00000000 00000001 |

Table 4.4: Polynomials used when creating  $K'_c$ .<sup>1</sup>

1. All polynomials are in hexadecimal notation. The LSB is in the rightmost position.

In Figure 4.5, all bits are shifted into the LFSRs, starting with the least significant bit (LSB). For instance, from the third octet of the address, BD\_ADDR[2], first BD\_ADDR<sub>16</sub> is entered, followed by BD\_ADDR<sub>17</sub>, etc. Furthermore, CL<sub>0</sub> corresponds to CLK<sub>1</sub>, ..., CL<sub>25</sub> corresponds to CLK<sub>26</sub>.

Note that the output symbols  $x_t^i$ ,  $i = 1, \dots, 4$  are taken from the positions 24, 24, 32, and 32 for LFSR<sub>1</sub>, LFSR<sub>2</sub>, LFSR<sub>3</sub>, and LFSR<sub>4</sub>, respectively (counting the leftmost position as number 1).

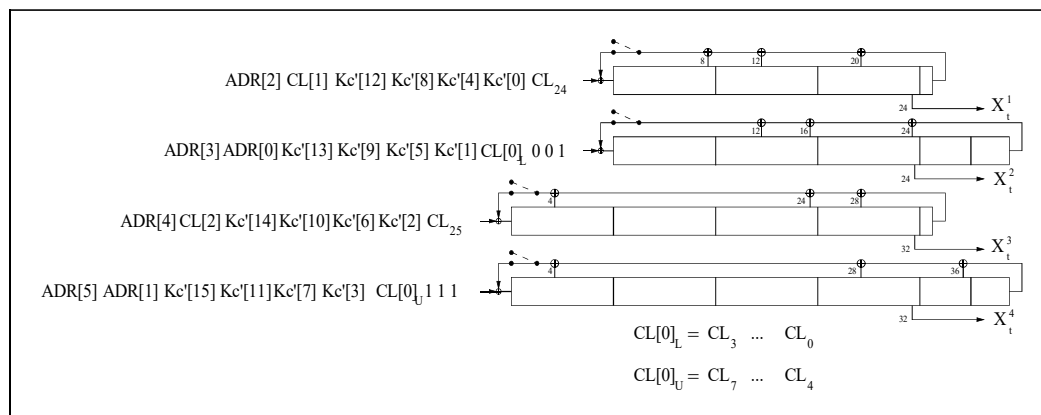


Figure 4.5: Arranging the input to the LFSRs.

In [Figure 4.6](#), the 128 binary output symbols  $Z_0, \dots, Z_{127}$  are arranged in octets denoted  $Z[0], \dots, Z[15]$ . The LSB of  $Z[0]$  corresponds to the first of these symbols, the MSB of  $Z[15]$  is the last output from the generator. These bits shall be loaded into the LFSRs according to the figure. It is a parallel load and no update of the blend registers is done. The first output symbol is generated at the same time. The octets shall be written into the registers with the LSB in the leftmost position (i.e. the opposite of before). For example,  $Z_{24}$  is loaded into position 1 of LFSR<sub>4</sub>.

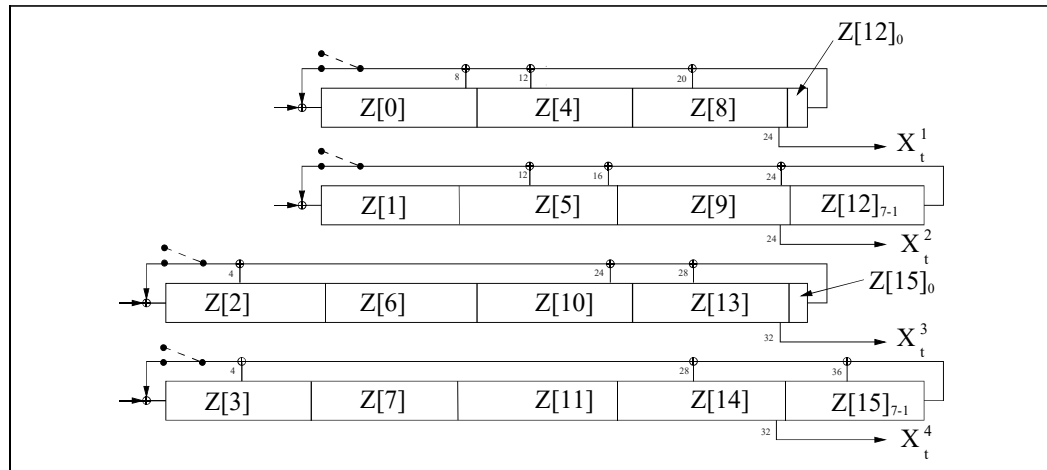


Figure 4.6: Distribution of the 128 last generated output symbols within the LFSRs.

## 4.6 KEY STREAM SEQUENCE

When the initialization is finished, the output from the summation combiner is used for encryption/decryption. The first bit to use shall be the one produced at the parallel load, i.e. at  $t = 240$ . The circuit shall be run for the entire length of the current payload. Then, before the reverse direction is started, the entire initialization process shall be repeated with updated values on the input parameters.

Sample data of the encryption output sequence can be found in [\[Part G\] Section 1 on page 953](#). A necessary, but not sufficient, condition for all Bluetooth compliant implementations of encryption is to produce these encryption streams for identical initialization values.

## 5 AUTHENTICATION

Authentication uses a challenge-response scheme in which a claimant's knowledge of a secret key is checked through a 2-move protocol using symmetric secret keys. The latter implies that a correct claimant/verifier pair share the same secret key, for example  $K$ . In the challenge-response scheme the verifier challenges the claimant to authenticate a random input (the challenge), denoted by  $AU\_RAND_A$ , with an authentication code, denoted by  $E_I$ , and return the result  $SRES$  to the verifier, see [Figure 5.1 on page 1082](#). This figure also shows that the input to  $E_I$  consists of the tuple  $AU\_RAND_A$  and the Bluetooth device address ( $BD\_ADDR$ ) of the claimant. The use of this address prevents a simple reflection attack<sup>1</sup>. The secret  $K$  shared by devices A and B is the current link key.

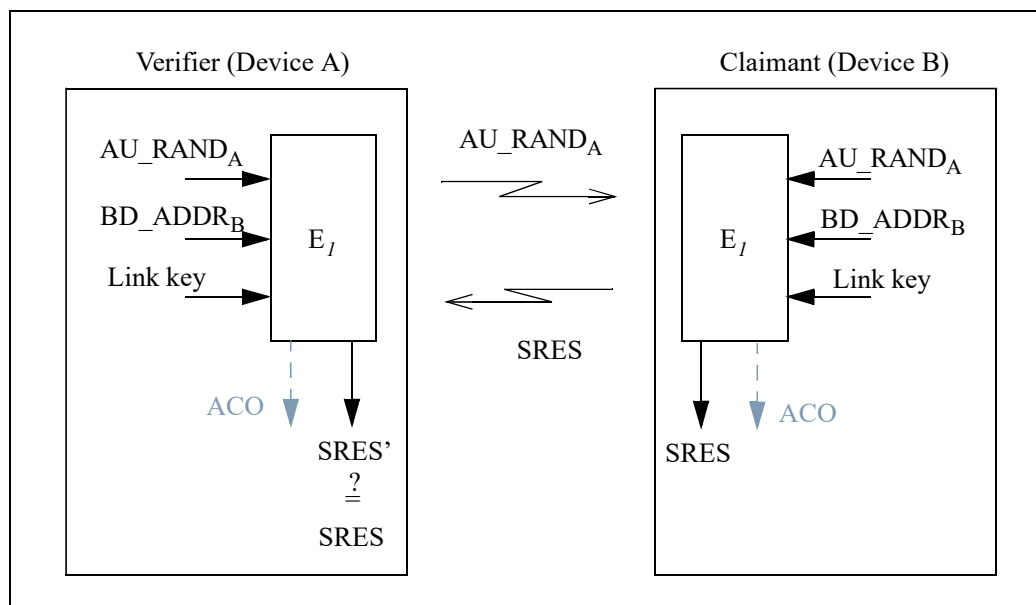


Figure 5.1: Challenge-response for the Bluetooth.

The challenge-response scheme for symmetric keys is depicted in [Figure 5.2 on page 1083](#).

1. The reflection attack actually forms no threat because all service requests are dealt with on a FIFO bases. When preemption is introduced, this attack is potentially dangerous.

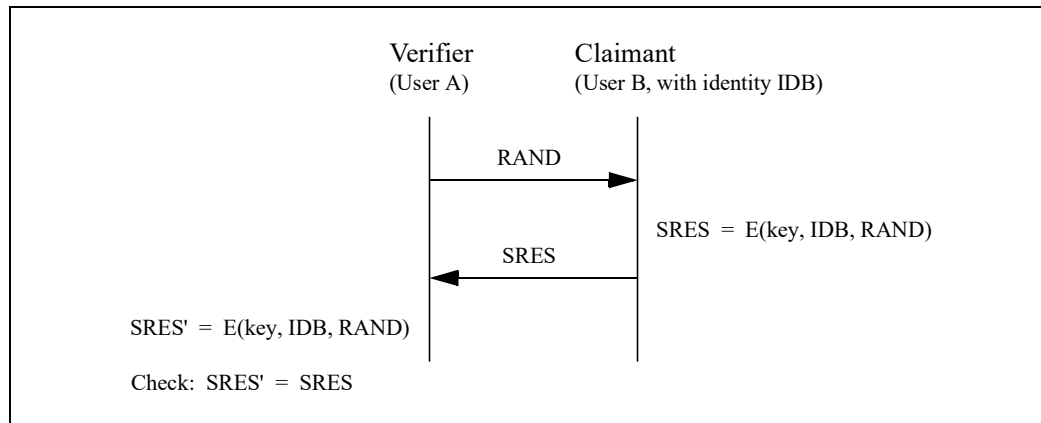


Figure 5.2: Challenge-response for symmetric key systems.

The verifier is not required to be the master. The application indicates which device has to be authenticated. Some applications only require a one-way authentication. However, some peer-to-peer communications, should use a mutual authentication in which each device is subsequently the challenger (verifier) in two authentication procedures. The LM shall process authentication preferences from the application to determine in which direction(s) the authentication(s) takes place. For mutual authentication with the devices of [Figure 5.1 on page 1082](#), after device A has successfully authenticated device B, device B could authenticate device A by sending an  $AU\_RAND_B$  (different from the  $AU\_RAND_A$  that device A issued) to device A, and deriving the SRES and SRES' from the new  $AU\_RAND_B$ , the address of device A, and the link key  $K_{AB}$ .

If an authentication is successful the value of ACO as produced by  $E_I$  shall be retained.

## 5.1 REPEATED ATTEMPTS

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<sup>1</sup>. 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 can use one of the following methods:

- Change its private key after three failed attempts from any BD\_ADDR and after 10 successful pairings from any BD\_ADDR; or after a combination of these such that 3 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 value larger than 1 may be used.

## 6 THE AUTHENTICATION AND KEY-GENERATING FUNCTIONS

This section describes the algorithms used for authentication and key generation.

### 6.1 THE AUTHENTICATION FUNCTION $E_I$

The authentication function  $E_I$  is a computationally secure authentication code  $E_I$ . uses the encryption function SAFER+. The algorithm is an enhanced version of an existing 64-bit block cipher SAFER-SK128, and it is freely available. In the following discussion, the block cipher will be denoted as the function  $A_r$  which maps using a 128-bit key, a 128-bit input to a 128-bit output, i.e.

$$A_r: \{0, 1\}^{128} \times \{0, 1\}^{128} \rightarrow \{0, 1\}^{128} \quad (EQ 11)$$

$$(k \times x) \mapsto t.$$

The details of  $A_r$  are given in the next section. The function  $E_I$  is constructed using  $A_r$  as follows

$$E_I: \{0, 1\}^{128} \times \{0, 1\}^{128} \times \{0, 1\}^{48} \rightarrow \{0, 1\}^{32} \times \{0, 1\}^{96} \quad (EQ 12)$$

$$(K, RAND, address) \mapsto (SRES, ACO),$$

where  $SRES = \text{Hash}(K, RAND, address, 6)[0, \dots, 3]$ , where Hash is a keyed hash function defined as<sup>1</sup>,

$$\text{Hash}: \{0, 1\}^{128} \times \{0, 1\}^{128} \times \{0, 1\}^{8 \times L} \times \{6, 12\} \rightarrow \{0, 1\}^{128} \quad (EQ 13)$$

$$(K, I_1, I_2, L) \mapsto A'_r([\tilde{K}], [E(I_2, L) +_{16} (A_r(K, I_1) \uparrow_{16} I_1)]),$$

and where

$$E: \{0, 1\}^{8 \times L} \times \{6, 12\} \rightarrow \{0, 1\}^{8 \times 16} \quad (EQ 14)$$

$$(X[0, \dots, L-1], L) \mapsto (X[i \pmod L] \text{ for } i = 0 \dots 15),$$

is an expansion of the  $L$  octet word  $X$  into a 128-bit word. The function  $A_r$  is evaluated twice for each evaluation of  $E_I$ . The key  $\tilde{K}$  for the second use of  $A_r$  (actually  $A'_r$ ) is offset from  $K$  as follows<sup>2</sup>

1. The operator  $+_{16}$  denotes bitwise addition mod 256 of the 16 octets, and the operator  $\oplus_{16}$  denotes bitwise XORing of the 16 octets.
2. The constants are the largest primes below 257 for which 10 is a primitive root.

$$\begin{aligned}
 \tilde{K}[0] &= (K[0] + 233) \bmod 256, & \tilde{K}[1] &= K[1] \oplus 229, \\
 \tilde{K}[2] &= (K[2] + 223) \bmod 256, & \tilde{K}[3] &= K[3] \oplus 193, \\
 \tilde{K}[4] &= (K[4] + 179) \bmod 256, & \tilde{K}[5] &= K[5] \oplus 167, \\
 \tilde{K}[6] &= (K[6] + 149) \bmod 256, & \tilde{K}[7] &= K[7] \oplus 131, \\
 \tilde{K}\{8\} &= K[8] \oplus 233, & \tilde{K}[9] &= (K[9] + 229) \bmod 256, \\
 \tilde{K}[10] &= K[10] \oplus 223, & \tilde{K}[11] &= (K[11] + 193) \bmod 256, \\
 \tilde{K}[12] &= K[12] \oplus 179, & \tilde{K}[13] &= (K[13] + 167) \bmod 256, \\
 \tilde{K}[14] &= K[14] \oplus 149, & \tilde{K}[15] &= (K[15] + 131) \bmod 256.
 \end{aligned}
 \tag{EQ 15}$$

A data flowchart of the computation of  $E_1$  is shown in [Figure 6.1 on page 1086](#).  $E_1$  is also used to deliver the parameter ACO (Authenticated Ciphering Offset) that is used in the generation of the ciphering key by  $E_3$ , see equations [\(EQ 3\) on page 1068](#) and [\(EQ 23\) on page 1091](#). The value of ACO is formed by the octets 4 through 15 of the output of the hash function defined in [\(EQ 13\) on page 1085](#):

$$ACO = \text{Hash}(K, \text{RAND}, \text{address}, 6)[4, \dots, 15]. \tag{EQ 16}$$

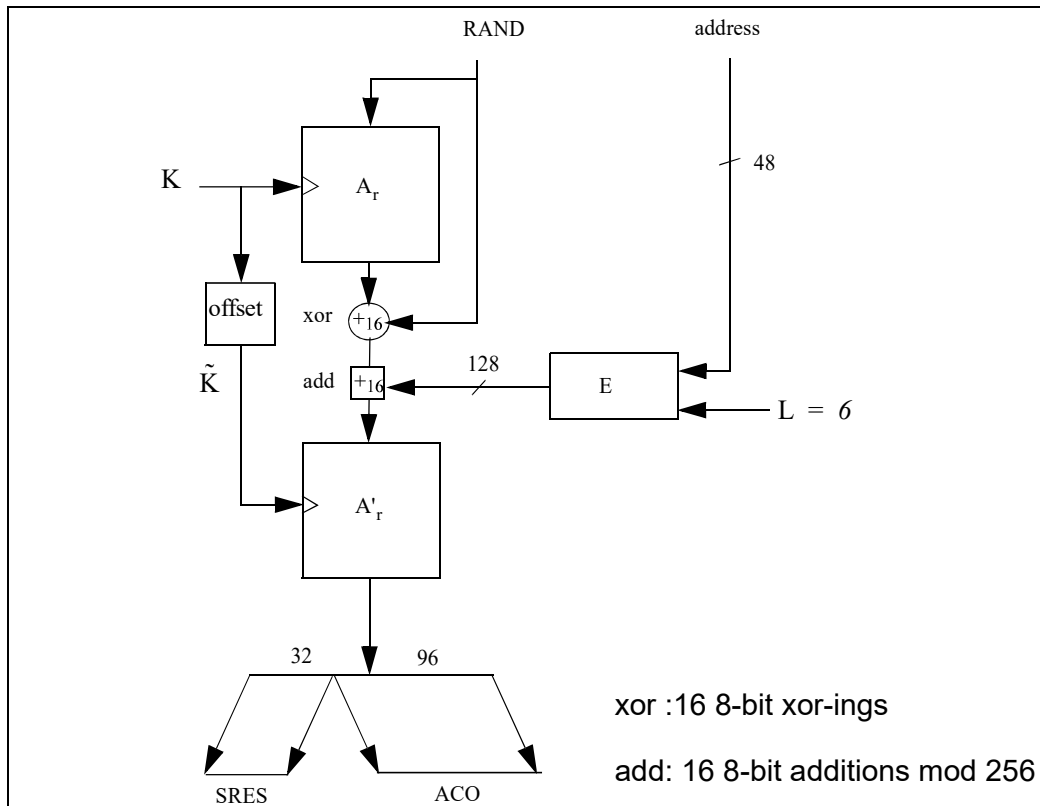


Figure 6.1: Flow of data for the computation of  $E_1$ .



## 6.2 THE FUNCTIONS $A_r$ AND $A'_r$

The function  $A_r$  is identical to SAFER+. It consists of a set of 8 layers, (each layer is called a round) and a parallel mechanism for generating the sub keys  $K_p[j]$ ,  $p = 1, 2, \dots, 17$ , which are the round keys to be used in each round. The function will produce a 128-bit result from a 128-bit random input string and a 128-bit key. Besides the function  $A_r$ , a slightly modified version referred to as  $A'_r$  is used in which the input of round 1 is added to the input of round 3. This is done to make the modified version non-invertible and prevents the use of  $A'_r$  (especially in  $E_{2x}$ ) as an encryption function. See [Figure 6.2 on page 1088](#) for details.

### 6.2.1 The round computations

The computations in each round are a composition of encryption with a round key, substitution, encryption with the next round key, and, finally, a Pseudo Hadamard Transform (PHT). The computations in a round shall be as shown in [Figure 6.2 on page 1088](#). The sub keys for round  $r$ ,  $r = 1, 2, \dots, 8$  are denoted  $K_{2r-1}[j]$ ,  $K_{2r}[j]$ ,  $j = 0, 1, \dots, 15$ . After the last round  $K_{17}[j]$  is applied identically to all previous odd numbered keys.

### 6.2.2 The substitution boxes “e” and “l”

In [Figure 6.2 on page 1088](#) two boxes are shown, marked “e” and “l”. These boxes implement the same substitutions as are used in SAFER+; i.e. they implement

$$\begin{aligned} e, l &: \{0, \dots, 255\} \rightarrow \{0, \dots, 255\}, \\ e &: i \mapsto (45^i \pmod{257}) \pmod{256}, \\ l &: i \mapsto j \text{ s.t. } i = e(j). \end{aligned}$$

Their role, as in the SAFER+ algorithm, is to introduce non-linearity.

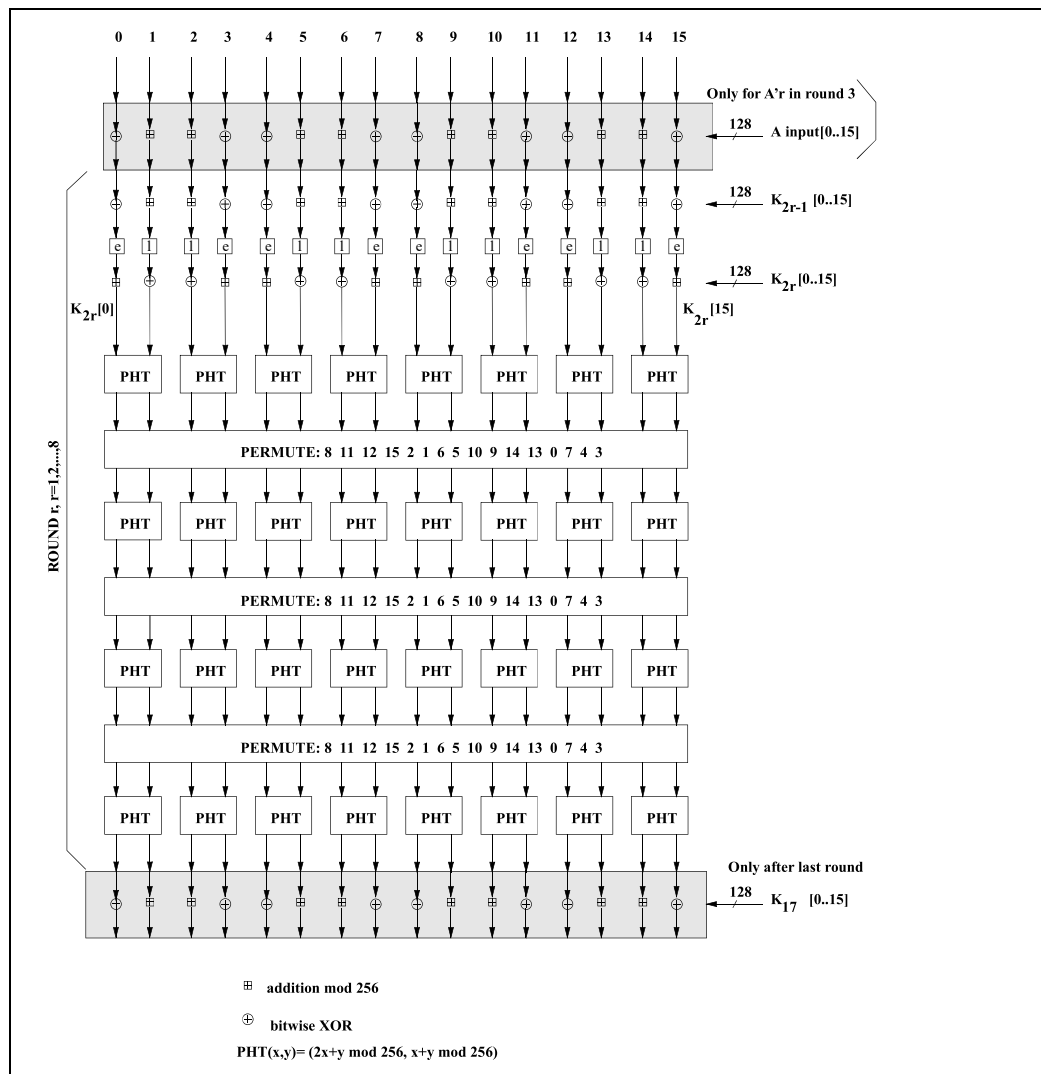


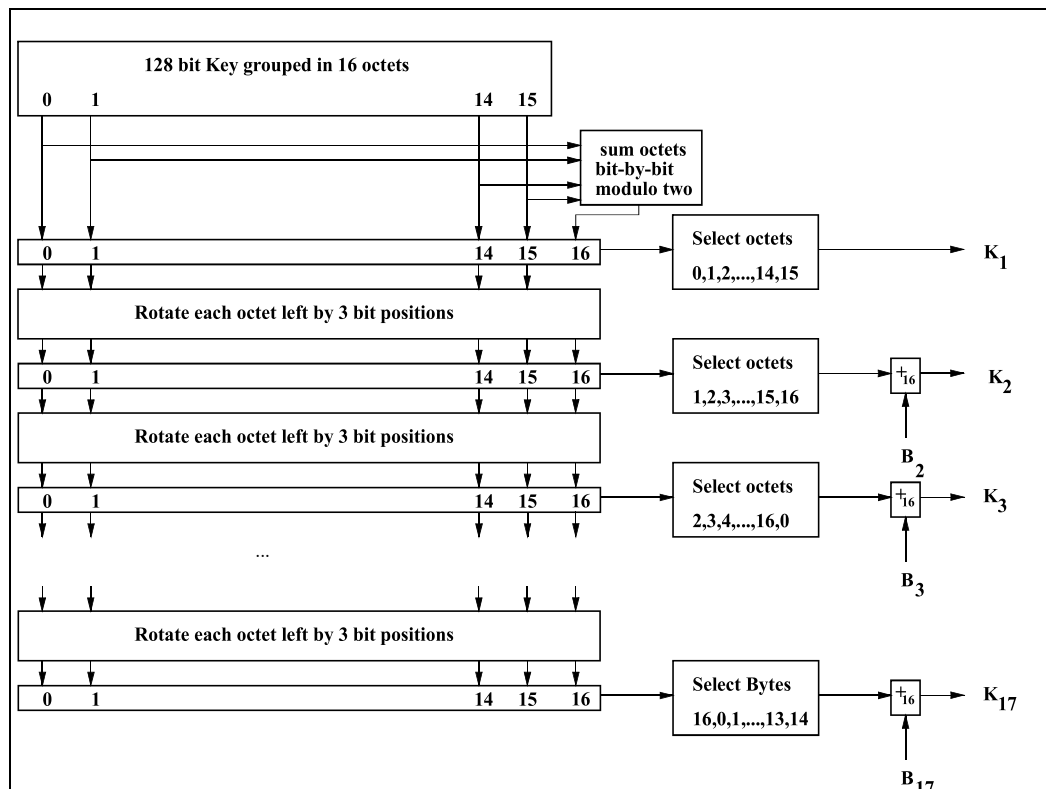
Figure 6.2: One round in  $A_r$  and  $A'_r$ .

In [Section 6.2](#), the permutation boxes show how input byte indices are mapped onto output byte indices. Thus, position 8 is mapped on position 0 (leftmost), position 11 is mapped on position 1, etc.

### 6.2.3 Key scheduling

In each round, 2 batches of 16 octet-wide keys are needed. These round keys are derived as specified by the key scheduling in SAFER+. [Figure 6.3 on page 1089](#) gives an overview of how the round keys  $K_p[j]$  are determined. The bias vectors  $B_2, B_3, \dots, B_{17}$  shall be computed according to following equation:

$$B_p[i] = ((45^{(45^{17p+i} \bmod 257)} \bmod 257) \bmod 256), \text{ for } i = 0, \dots, 15. \quad (\text{EQ 17})$$


 Figure 6.3: Key scheduling in  $A_p$ 

### 6.3 $E_2$ -KEY GENERATION FUNCTION FOR AUTHENTICATION

The key used for authentication shall be derived through the procedure that is shown in [Figure 6.4 on page 1091](#). The figure shows two modes of operation for the algorithm. In the first mode,  $E_{21}$  produces a 128-bit link key,  $K$ , using a 128-bit RAND value and a 48-bit address. This mode shall be utilized when creating unit keys and combination keys. In the second mode,  $E_{22}$  produces a 128-bit link key,  $K$ , using a 128-bit RAND value and an  $L$  octet user PIN. The second mode shall be used to create the initialization key, and also when a master key is to be generated.

When the initialization key is generated, the PIN is augmented with the BD\_ADDR, see [Section 3.2.1 on page 1066](#) for which address to use. The augmentation shall always start with the least significant octet of the address immediately following the most significant octet of the PIN. Since the maximum length of the PIN used in the algorithm cannot exceed 16 octets, it is possible that not all octets of BD\_ADDR will be used.



This key generating algorithm again exploits the cryptographic function  $E_2$ . for mode 1 (denoted  $E_{21}$ ) is computed according to following equations:

$$\begin{aligned} E_{21}: \{0, 1\}^{128} \times \{0, 1\}^{48} &\rightarrow \{0, 1\}^{128} \\ (\text{RAND}, \text{address}) &\mapsto A'_r(X, Y) \end{aligned} \quad (\text{EQ 18})$$

where (for mode 1)

$$\begin{cases} X = \text{RAND}[0 \dots 14] \cup (\text{RAND}[15] \oplus 6) \\ \quad \quad \quad 15 \\ Y = \bigcup_{i=0} \text{address}[i \pmod{6}] \end{cases} \quad (\text{EQ 19})$$

Let  $L$  be the number of octets in the user PIN. The augmenting is defined by

$$\text{PIN}' = \begin{cases} \text{PIN}[0 \dots L-1] \cup \text{BD\_ADDR}[0 \dots \min\{5, 15-L\}], & L < 16, \\ \text{PIN}[0 \dots L-1], & L = 16, \end{cases} \quad (\text{EQ 20})$$

Then, in mode 2,  $E_2$  (denoted  $E_{22}$ ) is

$$\begin{aligned} E_{22}: \{0, 1\}^{8L'} \times \{0, 1\}^{128} \times \{1, 2, \dots, 16\} &\rightarrow \{0, 1\}^{128} \\ (\text{PIN}', \text{RAND}, L') &\mapsto A'_r(X, Y) \end{aligned} \quad (\text{EQ 21})$$

where

$$\begin{cases} \quad \quad \quad 15 \\ X = \bigcup_{i=0} \text{PIN}'[i \pmod{L'}], \\ Y = \text{RAND}[0 \dots 14] \cup (\text{RAND}[15] \oplus L'), \end{cases} \quad (\text{EQ 22})$$

and  $L' = \min\{16, L+6\}$  is the number of octets in  $\text{PIN}'$ .

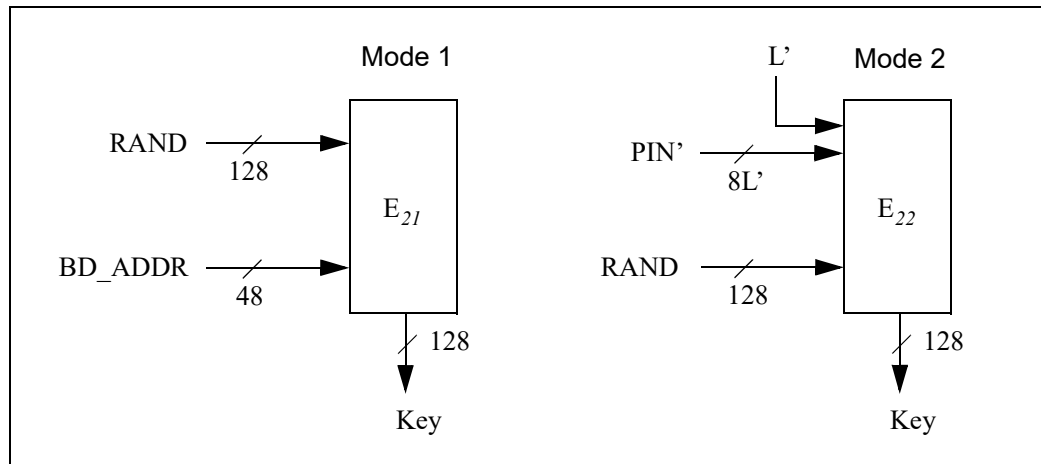


Figure 6.4: Key generating algorithm  $E_2$  and its two modes. Mode 1 is used for unit and combination keys, while mode 2 is used for  $K_{init}$  and  $K_{master}$

## 6.4 $E_3$ -KEY GENERATION FUNCTION FOR ENCRYPTION

The ciphering key  $K_C$  used by  $E_0$  shall be generated by  $E_3$ . The function  $E_3$  is constructed using  $A'_r$  as follows

$$E_3: \{0, 1\}^{128} \times \{0, 1\}^{128} \times \{0, 1\}^{96} \rightarrow \{0, 1\}^{128} \quad (\text{EQ 23})$$

$$(K, \text{RAND}, \text{COF}) \mapsto \text{Hash}(K, \text{RAND}, \text{COF}, 12)$$

where Hash is the hash function as defined by (EQ 13) on page 1085. The key length produced is 128 bits. However, before use within  $E_0$ , the encryption key  $K_C$  is shortened to the correct encryption key length, as described in Section 4.5 on page 1078. A block scheme of  $E_3$  is depicted in Figure 6.5.

The value of COF is determined as specified by equation (EQ 3) on page 1068.

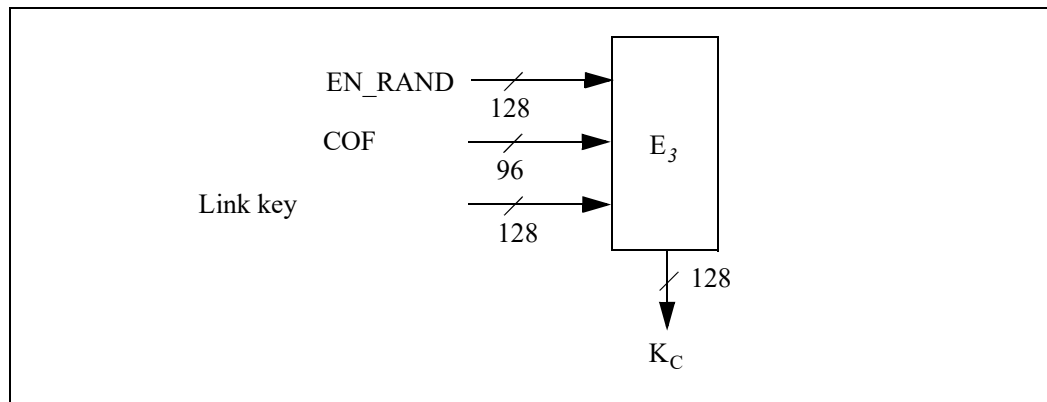


Figure 6.5: Generation of the encryption key

## 7 SECURE SIMPLE PAIRING

The Secure Simple Pairing security functions and procedures are described in this section. In addition, a cryptographic analysis of each procedure is provided.

There are five phases of Secure Simple Pairing:

- Phase 1: Public key exchange
- Phase 2: Authentication Stage 1
- Phase 3: Authentication Stage 2
- Phase 4: Link key calculation
- Phase 5: LMP Authentication and Encryption

Phases 1, 3, 4 and 5 are the same for all protocols whereas phase 2 (Authentication Stage 1) is different depending on the protocol used. Distributed through these five phases are 13 steps.

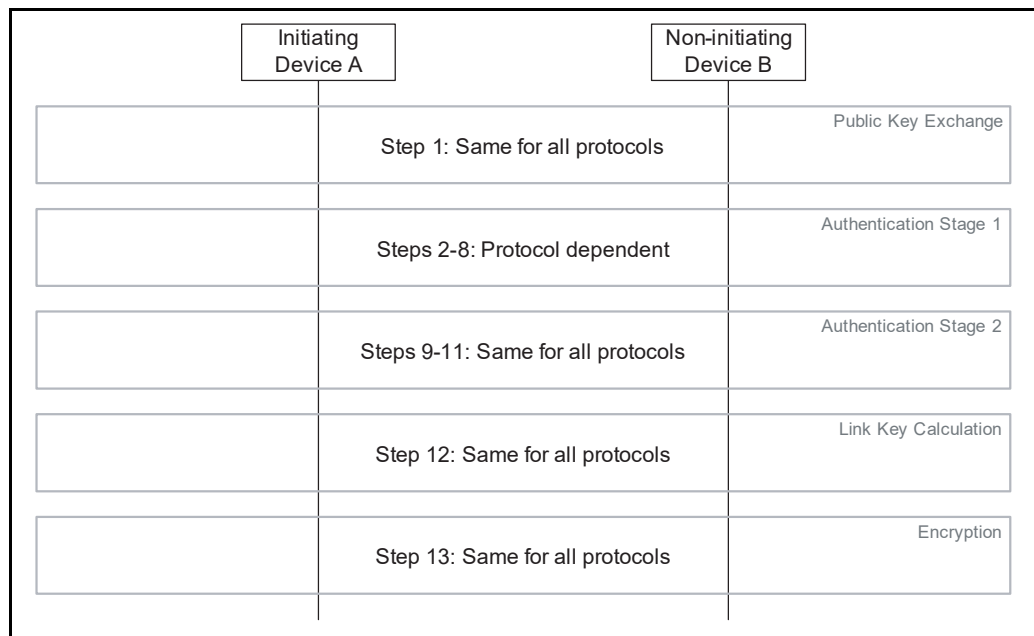


Figure 7.1: Simple Pairing Security Phases

The terminology used in the security sections is defined in [Table 7.1](#):

| Term | Definition                                                                              |
|------|-----------------------------------------------------------------------------------------|
| Cx   | Commitment value from device X                                                          |
| Cxi  | i <sup>th</sup> commitment value from device X. Only used in the passkey entry protocol |

Table 7.1: Terminology

| Term            | Definition                                                                                         |
|-----------------|----------------------------------------------------------------------------------------------------|
| DHKey           | Diffie Hellman key                                                                                 |
| Ex              | Check value from device X                                                                          |
| f1( )           | Used to generate the 128-bit commitment values Ca and Cb                                           |
| f2( )           | Used to compute the link key and possible other keys from the DHKey and random nonces              |
| f3( )           | Used to compute check values Ea and Eb in Authentication Stage 2                                   |
| g( )            | Used to compute numeric check values                                                               |
| h2( )           | Used to compute Generic AMP and Dedicated AMP keys                                                 |
| IOcapA          | IO capabilities of device A                                                                        |
| IOcapB          | IO capabilities of device B                                                                        |
| LK              | Link Key                                                                                           |
| Nx              | Nonce (unique random value) from device X                                                          |
| Nxi             | i <sup>th</sup> nonce (unique random value) from device X. Only used in the passkey entry protocol |
| PKx             | Public Key of device X                                                                             |
| rx              | Random value generated by device X                                                                 |
| rx <sub>i</sub> | Bit i of the random value rx. Only used in the passkey entry protocol                              |
| SKx             | Secret (Private) Key of device X                                                                   |
| Vx              | Confirmation value on device X. Only used in the numeric compare protocol.                         |
| X               | BD_ADDR of device X                                                                                |

Table 7.1: Terminology

## 7.1 PHASE 1: PUBLIC KEY EXCHANGE

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.

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.

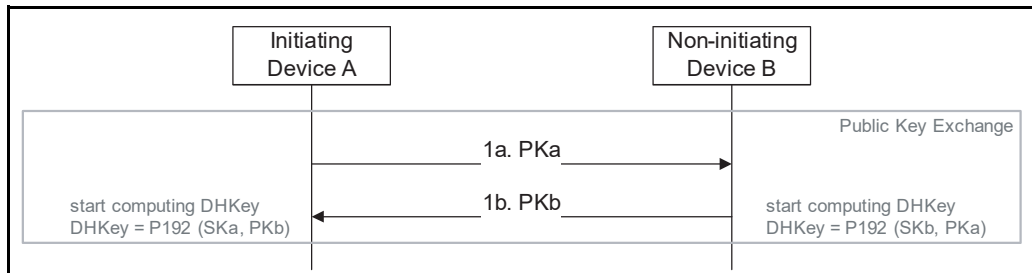


Figure 7.2: Public Key Exchange Details

## 7.2 PHASE 2: AUTHENTICATION STAGE 1

Authentication Stage 1 has three different protocols: Numeric Comparison, Out-of-Band, and Passkey Entry. Note that there is not a separate protocol for the Just Works association model. This association model shares the Numeric Comparison protocol.

The protocol is chosen based on the IO capabilities of the two devices.

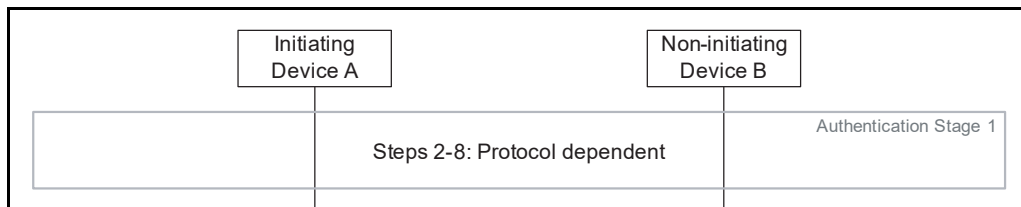


Figure 7.3: Authentication Stage 1 (High Level)

The three protocols are described in the following sections.

### 7.2.1 Authentication Stage 1: Numeric Comparison Protocol

The Numeric Comparison protocol provides limited protection against active "man-in-the-middle" (MITM) attacks as an active man-in-the-middle will succeed with a probability of 0.000001 on each iteration of the protocol. Provided that there is no MITM at the time the pairing is performed, the shared Link Key that is generated is computationally secure from even a passive eavesdropper that may have been present during the pairing.

The sequence diagram of Authentication Stage 1 for the Numeric Comparison protocol from the cryptographic point of view is shown below:

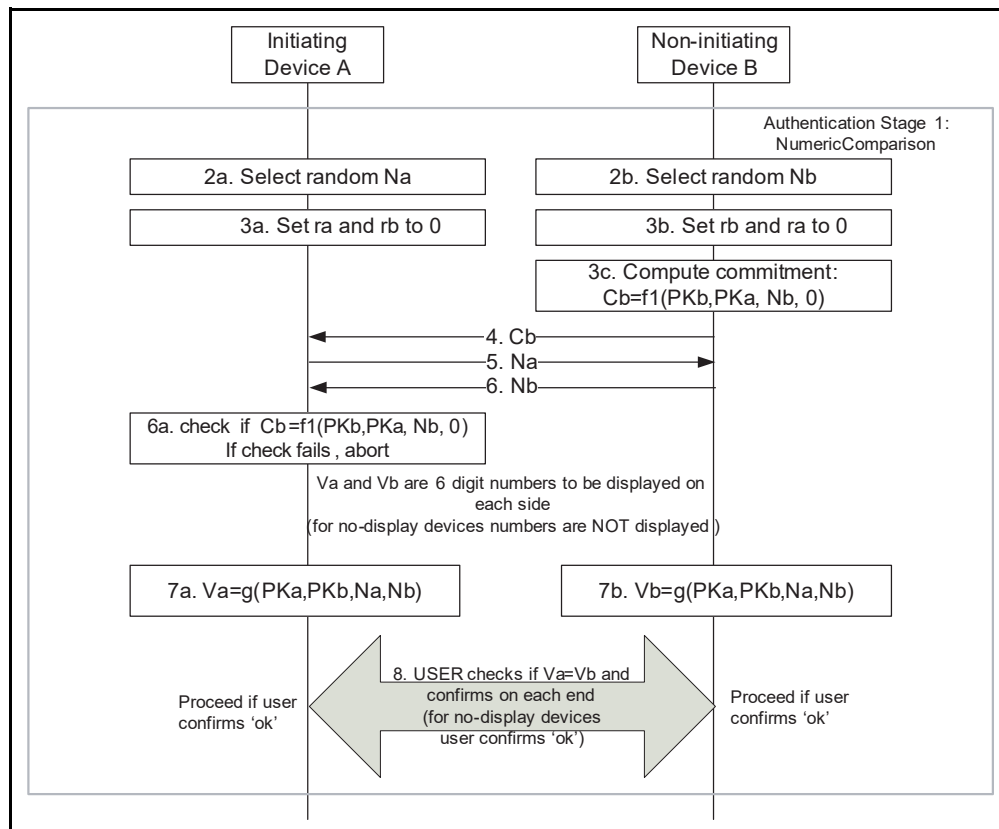


Figure 7.4: Authentication Stage 1: Numeric Comparison Protocol Details

After the public keys have been exchanged, each device selects a pseudo-random 128-bit nonce (step 2). This value is used to prevent replay attacks and must be freshly generated with each instantiation of the pairing protocol. This value should be generated directly from a physical source of randomness or with a good pseudo-random generator seeded with a random value from a physical source.

Following this the responding device then computes a commitment to the two public keys that have been exchanged and its own nonce value (step 3c). This commitment is computed as a one-way function of these values and is transmitted to the initiating device (step 4). The commitment prevents an attacker from changing these values at a later time.

The initiating and responding devices then exchange their respective nonce values (steps 5 and 6) and the initiating device confirms the commitment (step 6a). A failure at this point indicates the presence of an attacker or other transmission error and causes the protocol to abort. The protocol may be repeated with or without the generation of new public-private key pairs, but new nonces must be generated if the protocol is repeated.

Assuming that the commitment check succeeds, the two devices each compute 6-digit confirmation values that are displayed to the user on their respective devices (steps 7a, 7b, and 8). The user is expected to check that these 6-



digit values match and to confirm if there is a match. If there is no match, the protocol aborts and, as before, new nonces must be generated if the protocol is to be repeated.

An active MITM must inject its own key material into this process to have any effect other than denial-of-service. A simple MITM attack will result in the two 6-digit display values being different with probability 0.999999. A more sophisticated attack may attempt to engineer the display values to match, but this is thwarted by the commitment sequence. If the attacker first exchanges nonces with the responding device, it must commit to the nonce that it will use with the initiating device before it sees the nonce from the initiating device. If the attacker first exchanges nonces with the initiating device, it must send a nonce to the responding device before seeing the nonce from the responding device. In each case, the attacker must commit to at least the second of its nonces before knowing the second nonce from the legitimate devices. It therefore cannot choose its own nonces in such a way as to cause the display values to match.

### 7.2.2 Authentication Stage 1: Out of Band Protocol

The Out-of-Band protocol is used when authentication information has been received by at least one of the devices and indicated in the OOB Authentication Data Present parameter in the LMP IO capability exchange sequence. The mode in which the discovery of the peer device is first done in-band and then followed by the transmission of authentication parameters through OOB interface is not supported. The sequence diagram for Authentication Stage 1 for Out of Band from the cryptographic point of view is shown below:

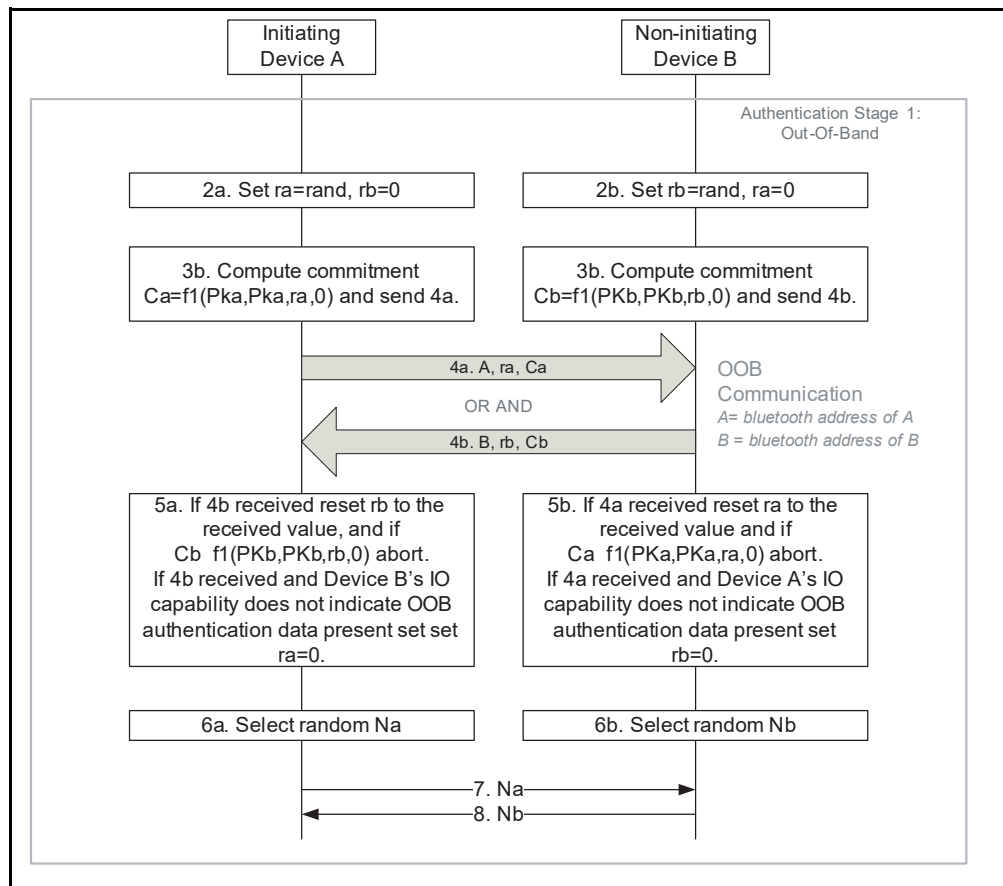


Figure 7.5: Authentication Stage 1: Out of Band Details

**Principle of operation:** If both devices can transmit and/or receive data over an out-of-band channel, then mutual authentication will be based on the commitments of the public keys ( $C_a$  and  $C_b$ ) exchanged OOB in Authentication stage 1. If OOB communication is possible only in one direction (e.g., one device can only be read as e.g. is the case of a device equipped with a passive NFC tag), then authentication of the device receiving the OOB communication will be based on that device knowing a random number  $r$  sent via OOB. In this case,  $r$  must be secret:  $r$  can be created afresh every time, or access to the device sending  $r$  must be restricted. If  $r$  is not sent by a device, it is assumed to be 0 by the device receiving the OOB information in step 4a or 4b.

If the OOB communication cannot be tampered with (i.e. immune to a MITM attack), this association method is also not susceptible to MITM attack. Also, in the Out of Band protocol the size of authentication parameters ( $C_a$ ,  $C_b$  and  $r_a$ ,  $r_b$ ) need not be restricted by what the user can comfortably read or type. For that reason the Out of Band protocol can be more secure than using the Numeric Comparison or Passkey Entry protocols. However, both devices need to have matching OOB interfaces.

**Roles of A and B:** The OOB Authentication Stage 1 protocol is symmetric with respect to the roles of A and B. It does not require that device A always will ini-



tiate pairing and it automatically resolves asymmetry in the OOB communication, e.g. in the case when one of the devices has an NFC tag and can only transmit OOB.

**Order of steps:** The public key exchange must happen before the verification step 5. In the diagram the in-band public key exchange between the devices (step 1) is done before the OOB communication (step 4). But when the pairing is initiated by an OOB interface, public key exchange will happen after the OOB communication (step 1 will be between steps 4 and 5).

Values of  $r_a$  and  $r_b$ : Since the direction of the peer's OOB interface cannot be verified before the OOB communication takes place, a device should always generate and if possible transmit through its OOB interface a random number  $r$  to the peer. Each device applies the following rules locally to set the values of its own  $r$  and the value of the peer's  $r$ :

1. Initially,  $r$  of the device is set to a random number and  $r$  of the peer is set to 0 (step 2).
2. If a device has received OOB, it sets the peer's  $r$  value to what was sent by the peer (Step 5).
3. If the remote device's OOB Authentication Data parameter sent in the LMP IO capabilities exchange sequence is set to No OOB Data Received, it sets its own  $r$  value to 0 (Step 5)

These rules ensure that when entering Authentication Stage 2, both A and B have the same values for  $r_a$  and  $r_b$  if OOB communication took place.

### 7.2.3 Authentication Stage 1: Passkey Entry Protocol

The Passkey Entry protocol is used when LMP IO capability exchange sequence indicates that Passkey Entry shall be used.

The sequence diagram for Authentication Stage 1 for Passkey Entry from the cryptographic point of view is shown in [Figure 7.6](#).

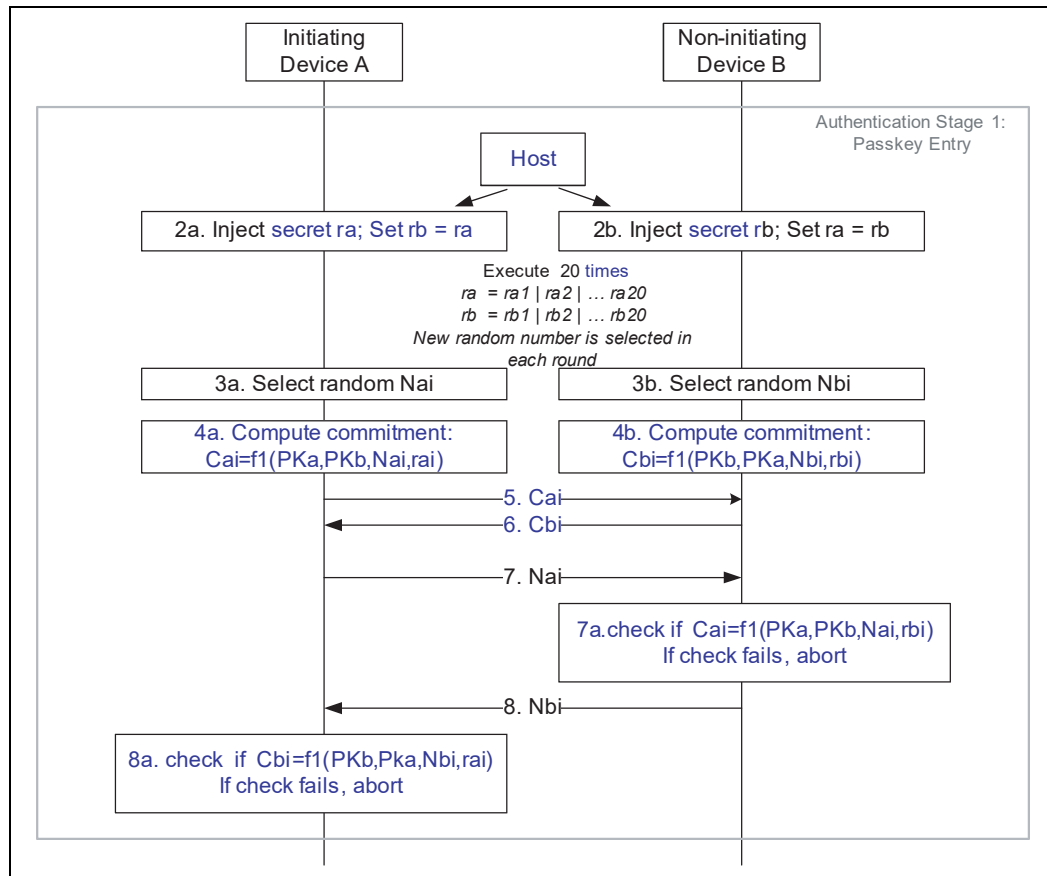


Figure 7.6: Authentication Stage 1: Passkey Entry Details

The user inputs an identical Passkey into both devices. Alternately, the Passkey may be generated and displayed on one device, and the user then inputs it into the other (step 2). This short shared key will be the basis of the mutual authentication of the devices. Steps 3 through 8 are repeated  $k$  times for a  $k$ -bit Passkey --e.g.,  $k=20$  for a 6-digit Passkey ( $999999=0xF423F$ ).

In Steps 3-8, each side commits to each bit of the Passkey, using a long nonce (128 bits), and sending the hash of the nonce, the bit of the Passkey, and both public keys to the other party. The parties then take turns revealing their commitments until the entire Passkey has been mutually disclosed. The first party to reveal a commitment for a given bit of the Passkey effectively reveals that bit of the Passkey in the process, but the other party then has to reveal the corresponding commitment to show the same bit value for that bit of the Passkey, or else the first party will then abort the protocol, after which no more bits of the Passkey are revealed.

This "gradual disclosure" prevents leakage of more than 1 bit of un-guessed Passkey information in the case of a MITM attack. A MITM attacker with only partial knowledge of the Passkey will only receive one incorrectly-guessed bit of the Passkey before the protocol fails. Hence, a MITM attacker who engages

first one side, then the other will only gain an advantage of at most two bits over a simple brute-force guesser which succeeds with probability 0.000001.

The long nonce is included in the commitment hash to make it difficult to brute-force even after the protocol has failed. The public Diffie-Hellman values are included to tie the Passkey protocol to the original ECDH key exchange, to prevent a MITM from substituting the attacker's public key on both sides of the ECDH exchange in standard MITM fashion.

At the end of this stage,  $N_a$  is set to  $N_{a20}$  and  $N_b$  is set to  $N_{b20}$  for use in Authentication Stage 2.

### 7.3 PHASE 3: AUTHENTICATION STAGE 2

The second stage of authentication then confirms that both devices have successfully completed the exchange. This stage is identical in all three protocols.

Each device computes a new confirmation value that includes the previously exchanged values and the newly derived shared key (step 9). The initiating device then transmits its confirmation value which is checked by the responding device (step 10). If this check fails, it indicates that the initiating device has not confirmed the pairing, and the protocol must be aborted. The responding device then transmits its confirmation value which is checked by the initiating device (step 11). A failure indicates that the responding device has not confirmed the pairing and the protocol should abort.

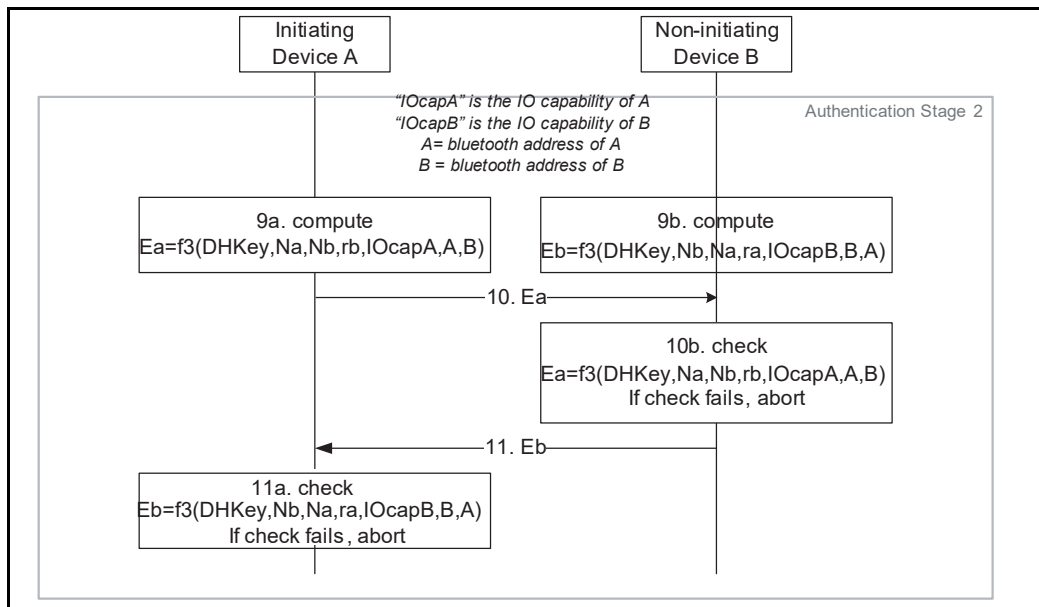


Figure 7.7: Authentication Stage 2

## 7.4 PHASE 4: LINK KEY CALCULATION

Once both sides have confirmed the pairing, a link key is computed from the derived shared key and the publicly exchanged data (step 12). The nonces ensure the freshness of the key even if long-term ECDH values are used by both sides. This link key is used to maintain the pairing.

When computing the link key both parties shall input the parameters in the same order to ensure that both devices compute the same key: device A's parameters are those of the piconet master and device B's parameters are those of the piconet slave.

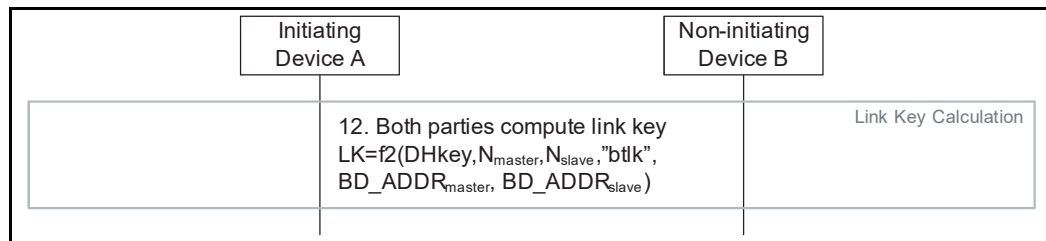


Figure 7.8: Link Key Calculation

## 7.5 PHASE 5: LMP AUTHENTICATION AND ENCRYPTION

The final phase in Simple Pairing is generating the encryption keys. This is the same as the final steps in legacy pairing.

## 7.6 ELLIPTIC CURVE DEFINITION

Simple Pairing uses the FIPS P-192 curve defined in [13]. Elliptic curves are specified by  $p$ ,  $a$ ,  $b$  and are of the form

$$E: y^2 = x^3 + ax + b \pmod{p}$$

For each value of  $b$  a unique curve can be developed. In NIST P-192:

$$a = \text{mod}(-3, p)$$

$b$  is defined and its method of generation can be verified by using SHA-1 (with a given seed  $s$  and using  $b^2c = -27 \pmod{p}$ )

The following parameters are given:

- The prime modulus  $p$ , order  $r$ , base point  $x$ -coordinate  $G_x$ , base point  $y$ -coordinate  $G_y$ .
- 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  $u^{\text{th}}$  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  $2^{192}/2$ ).

## 7.7 CRYPTOGRAPHIC FUNCTION DEFINITIONS

In addition to computing the Elliptic Curve Diffie Hellman key, the Numeric Comparison, Out-of-Band and Passkey Entry protocols require four cryptographic functions. These functions are known as  $f1$ ,  $g$ ,  $f2$  and  $f3$ .

$f1$  is used to generate the 128-bit commitment values  $C_a$  and  $C_b$

$g$  is used to compute the 6-digit numeric check values

$f2$  is used to compute the link key and possible other keys from the DHKey and random nonces

$f3$  is used to compute check values  $E_a$  and  $E_b$  in Authentication Stage 2.

The basic building block for these functions is based on as it already exists in the key derivation function E3 (which is the same as E1 but with a larger input).

Inside the  $f1$ ,  $g$ ,  $f2$ , and  $f3$  cryptographic functions, when a multi-octet integer input parameter is used as input to the SHA-256 and HMAC-SHA-256 functions, the most significant octet of the integer parameter shall be the first octet of the stream and the least significant octet of the integer parameter shall be the last octet of the stream. The output of the  $f1$ ,  $g$ ,  $f2$  and  $f3$  cryptographic functions is a multi-octet integer where the first octet out of SHA-256 and HMAC-SHA-256 shall be the MSB and the last octet shall be the LSB of that parameter.

Inside the  $h2$  cryptographic function, when a multi-octet integer input parameter is used as input to the SHA-256 and HMAC-SHA-256 functions, the most significant octet of the integer parameter shall be the last octet of the stream and the least significant octet of the integer parameter shall be the first octet of the stream. The output of the  $h2$  cryptographic function is a multi-octet integer where the first octet out of SHA-256 and HMAC-SHA-256 shall be the LSB and the last octet shall be the MSB of that parameter.

### 7.7.1 The Simple Pairing Commitment Function $f1$

The commitments are computed with function  $f1$ . The definition of the Simple Pairing commitment function makes use of the MAC function HMAC based on  $\text{SHA-256}_x$ , which is denoted as  $\text{HMAC-SHA-256}_x$  with 128-bit key  $X$ .



The inputs to the Simple Pairing function  $f1$  are:

U is 192 bits

V is 192 bits

X is 128 bits

Z is 8 bits

Z is zero (i.e. 8 bits of zeros) for Numeric Comparison and OOB protocol. In the Passkey protocol the most significant bit of Z is set equal to one and the least significant bit is made up from one bit of the passkey e.g. if passkey is '1' then  $Z = 0x81$  and if the passkey is '0' then  $Z = 0x80$ .

The output of the Simple Pairing  $f1$  function is:

$$f1(U, V, X, Z) = \text{HMAC-SHA-256}_X (U \parallel V \parallel Z) / 2^{128}$$

The inputs to  $f1$  are different depending on the different protocols.

| Numeric Comparison                                | Out-Of-Band                                       | Passkey Entry                                                       |
|---------------------------------------------------|---------------------------------------------------|---------------------------------------------------------------------|
| $Ca = f1(\text{PKax}, \text{PKbx}, \text{Na}, 0)$ | $Ca = f1(\text{PKax}, \text{PKax}, \text{Ra}, 0)$ | $\text{Cai} = f1(\text{PKax}, \text{PKbx}, \text{Nai}, \text{rai})$ |
| $Cb = f1(\text{PKbx}, \text{PKax}, \text{Nb}, 0)$ | $Cb = f1(\text{PKbx}, \text{PKbx}, \text{Rb}, 0)$ | $\text{Cbi} = f1(\text{PKbx}, \text{PKax}, \text{Nbi}, \text{rbi})$ |

Table 7.2: Inputs to  $f1$  for the different protocols

where PKax denotes the x-coordinate of the public key PKa of A. Similarly, PKbx denotes the x-coordinate of the public key PKb of B. Nai is the nonce value of  $i^{\text{th}}$  round. For each round Nai value is a new 128 bit number. Similarly, rai is one bit value of the passkey expanded to 8 bits (either 0x80 or 0x81).

### 7.7.2 The Simple Pairing Numeric Verification Function $g$

The Simple Pairing  $g$  function is defined as follows:

The inputs to the Simple Pairing function  $g$  are:

U is 192 bits

V is 192 bits

X is 128 bits

Y is 128 bits

The output of the Simple Pairing  $g$  function is:

$$g(U, V, X, Y) = \text{SHA-256}(U \parallel V \parallel X \parallel Y) \bmod 2^{32}$$

The numeric verification value is taken as six least significant digits of the 32-bit integer  $g(\text{PKax}, \text{PKbx}, \text{Na}, \text{Nb})$  where PKax denotes the x-coordinate of the



public key PK<sub>a</sub> of A and PK<sub>b</sub> denotes the x-coordinate of the public key PK<sub>b</sub> of B.

Output of SHA-256 is truncated to 32 bits by taking the least significant 32 bits of the output of SHA-256. This value is converted to decimal numeric value. The checksum used for numeric comparison is the least significant six digits.

$$\text{Compare Value} = g(U, V, X, Y) \bmod 10^6$$

For example, if output = 0x 01 2e b7 2a then decimal value = 19838762 and the checksum used for numeric comparison is 838762.

### 7.7.3 The Simple Pairing Key Derivation Function $f_2$

The definition of the Simple Pairing key derivation function makes use of the MAC function HMAC based on SHA-256, which is denoted as HMAC-SHA-256<sub>W</sub> with 192-bit key W.

The inputs to the Simple Pairing function  $f_2$  are:

W is 192 bits

N<sub>1</sub> is 128 bits

N<sub>2</sub> is 128 bits

keyID is 32 bits

A<sub>1</sub> is 48 bits

A<sub>2</sub> is 48 bits

The string "btlk" is mapped into keyID using extended ASCII<sup>1</sup> as follows:

keyID[0] = 0110 1011 (1sb)

keyID[1] = 0110 1100

keyID[2] = 0111 0100

keyID[3] = 0110 0010

KeyID = 0x62746c6b

The output of the Simple Pairing  $f_2$  function is:

$$f_2(W, N_1, N_2, \text{KeyID}, A_1, A_2) = \text{HMAC-SHA-256}_W(N_1 \parallel N_2 \parallel \text{KeyID} \parallel A_1 \parallel A_2) / 2^{128}$$

The output of  $f_2$  is taken as the 128 most significant (leftmost) bits of the output of HMAC-SHA-256.

1. ISO/IEC 8859-1  
(see [www.iso.org](http://www.iso.org))



The link key is then calculated as:

$$LK = f2(DHKey, N\_master, N\_slave, "btlk", BD\_ADDR\_master, BD\_ADDR\_slave)$$

#### 7.7.4 The Simple Pairing Check Function f3

The definition of the Simple Pairing f3 check function makes use of the MAC function HMAC based on SHA-256, which is denoted as HMAC-SHA-256<sub>W</sub> with 192-bit key W.

The inputs to the Simple Pairing function f3 are

W is 192 bits

N<sub>1</sub> is 128 bits

N<sub>2</sub> is 128 bits

R is 128 bits

IOcap is 24 bits

A<sub>1</sub> is 48 bits

A<sub>2</sub> is 48 bits

IOcap is three octets with the most significant octet as the Authentication Requirements parameter, the middle octet as the LMP Out-of-Band Authentication Data parameter, and the least significant octet as the LMP IO capability parameter.

The output of the Simple Pairing f3 function is:

$$f3(W, N_1, N_2, R, IOcap, A_1, A_2) = \text{HMAC-SHA-256}_W(N_1 \parallel N_2 \parallel R \parallel IOcap \parallel A_1 \parallel A_2) / 2^{128}$$

The output of f3 is taken as the 128 most significant (leftmost) bits of the output of HMAC-SHA-256. The check values are computed with function f3. The inputs to f3 are different depending on the different protocols:

| Numeric Comparison                      | Out-Of-Band                              | Passkey Entry                                |
|-----------------------------------------|------------------------------------------|----------------------------------------------|
| Ea = f3(DHKey, Na, Nb, 0, IOcapA, A, B) | Ea = f3(DHKey, Na, Nb, rb, IOcapA, A, B) | Ea = f3(DHKey, Na20, Nb20, rb, IOcapA, A, B) |
| Eb = f3(DHKey, Nb, Na, 0, IOcapB, B, A) | Eb = f3(DHKey, Nb, Na, ra, IOcapB, B, A) | Eb = f3(DHKey, Nb20, Na20, ra, IOcapB, B, A) |

Table 7.3: Inputs to f3 for the different protocols

DHKey denotes the shared secret Diffie-Hellman Key computed as P192(SKa, PKb) by A and as P192(SKb, PKa) by B. Adata denotes the capability data of A and Bdata denotes the capability data of B. In Passkey Entry, the data ra and rb are 6-digit passkey values which are expressed as a 128-bit integer. For instance,

if the 6-digit value of ra is 131313



then  $R = 0x\ 00\ 00\ 00\ 00\ 00\ 00\ 00\ 00\ 00\ 00\ 00\ 00\ 00\ 02\ 00\ f1$ .

The input A is the BD\_ADDR of device A and the input B is the BD\_ADDR of device B.

### 7.7.5 The Simple Pairing AMP Key Derivation Function h2

The definition of the Simple Pairing dedicated AMP key derivation function makes use of the MAC function HMAC based on SHA-256, which is denoted as HMAC-SHA-256<sub>W</sub> with 256-bit key W.

The inputs to the Simple Pairing function h2 are as follows:

W is 256 bits  
keyID is 32 bits  
L is 8 bits

The output of the Simple Pairing h2 function is as follows:

$$h2(W, \text{keyID}, L) = \text{HMAC-SHA-256}_W(\text{keyID}) / 2^{256-(L*8)}$$

For each Dedicated AMP, a fixed value of the key identifier KeyID is specified. A master key MK<sub>AMP</sub> of length L<sub>AMP</sub>,  $0 \leq L_{AMP} \leq 32$ , is computed as

$$\text{MK}_{AMP} = h2(\text{GAMP\_LK}, \text{keyID}_{AMP}, L_{AMP}).$$

#### 7.7.5.1 Initial GAMP\_LK Creation

The Generic AMP Link Key (GAMP\_LK) shall be created every time the link key (LK) is changed or created using the f2 function (see [Section 7.7.3](#)). Whenever the GAMP\_LK is created, all dedicated AMP link keys associated with the remote BD\_ADDR shall be discarded.

A fixed value of the key identifier KeyID is specified ('gamp'). The string 'gamp' is mapped into keyID using extended ASCII<sup>1</sup> as follows:

keyID[0] = 0111 0000 (lsb)  
keyID[1] = 0110 1101  
keyID[2] = 0110 0001  
keyID[3] = 0110 0111  
keyID = 0x67616d70

The initial GAMP\_LK shall be created in the Host using the h2 function.

$$\text{GAMP\_LK} = h2(W, \text{'gamp'}, 32)$$

Where  $W = \text{LK} || \text{LK}$

1. (see [www.iso.org](http://www.iso.org))ISO\_8859-1



### 7.7.5.2 Dedicated AMP Link Key Generation

Each AMP uses a defined keyID and keyLength. Both values are defined in the [Assigned numbers page](#).

A dedicated AMP link key is computed as

$$\text{Dedicated\_AMP\_Link\_Key} = h2(\text{GAMP\_LK}, \text{keyID}, \text{keyLength})$$

### 7.7.5.3 GAMP\_LK Regeneration

Each time the GAMP\_LK is used to generate a valid dedicated AMP link key it shall be regenerated by computing

$$\text{New\_GAMP\_LK} = h2(\text{GAMP\_LK}, \text{'gamp'}, 32)$$

and then setting

$$\text{GAMP\_LK} = \text{New\_GAMP\_LK}$$

### 7.7.5.4 Debug AMP Link Key Generation

When the BR/EDR link key is a debug link key, the dedicated AMP link keys are set equal to the keyLength least significant octets of the Generic AMP link key. When debug link keys are used, the Generic AMP link key shall not be regenerated.

## 8 AMP SECURITY

### 8.1 CREATION OF THE INITIAL GENERIC AMP LINK KEY

AMP security starts any time the BR/EDR link key is created or changed. This may be as the result of the f2 function during Phase 4 of the Secure Simple Pairing process (see [Section 7.7.3](#)) or due to a change connection link key procedure. The initial Generic AMP link key, GAMP\_LK, is created with the h2 function using the BR/EDR Link Key concatenated together with itself to form a 256-bit value and with the KeyID 'gamp' (see [Section 7.7.5.1](#)). The Generic AMP Link Key (GAMP\_LK) is stored in the security database alongside the BR/EDR Link Key once pairing has completed.

Whenever the Generic AMP Link Key is created, all Dedicated AMP Link Keys associated with the remote device shall be discarded and any active AMP Physical Links to the remote device shall be disconnected. Note that when a Host initiates a Change Connection Link Key procedure, all active AMP physical links between the two devices will be disconnected.

Since AMP security does not affect the BR/EDR Link Key, backwards compatibility is maintained for devices that support the Generic AMP feature with devices that do not. The Generic AMP Link Key is generated as part of Secure Simple Pairing regardless of whether the devices both support the Generic AMP feature or any AMP in common.

### 8.2 CREATION OF DEDICATED AMP LINK KEYS

When an AMP is used for the first time, a dedicated AMP key is created by the AMP Manager for that AMP type using the new Secure Simple Pairing function h2 and a KeyID for the AMP type. The length of the Dedicated AMP Link Key depends on the AMP Type. If the Dedicated AMP Link Key already exists due to a previous connection the AMP link key is not created and the stored key is reused.

The Dedicated AMP Link Key is sent down to the PAL in the *HCI Create Physical Link* and *HCI Accept Physical Link* commands (see [\[Vol. 2\], Part E Section 7.1.37](#) and [\[Vol. 2\], Part E Section 7.1.38](#)). Each PAL is responsible for establishing security from the Dedicated AMP Link Key. Note that a Dedicated AMP Link Key may be used for multiple sessions over the same AMP.

Each time a dedicated AMP key is created and the AMP Create Physical Link sequence completes successfully (see [\[Vol. 3\], Part E Section 2.3.3](#)), the Generic AMP Link Key is updated. This is performed using the h2 function with KeyID 'gamp.' If the AMP Create Physical Link sequence does not complete successfully, the dedicated AMP link key shall be discarded and the Generic AMP Link Key shall not be updated.

**Figure 8.1** The following figure shows an overview of the AMP key generation in the Host.

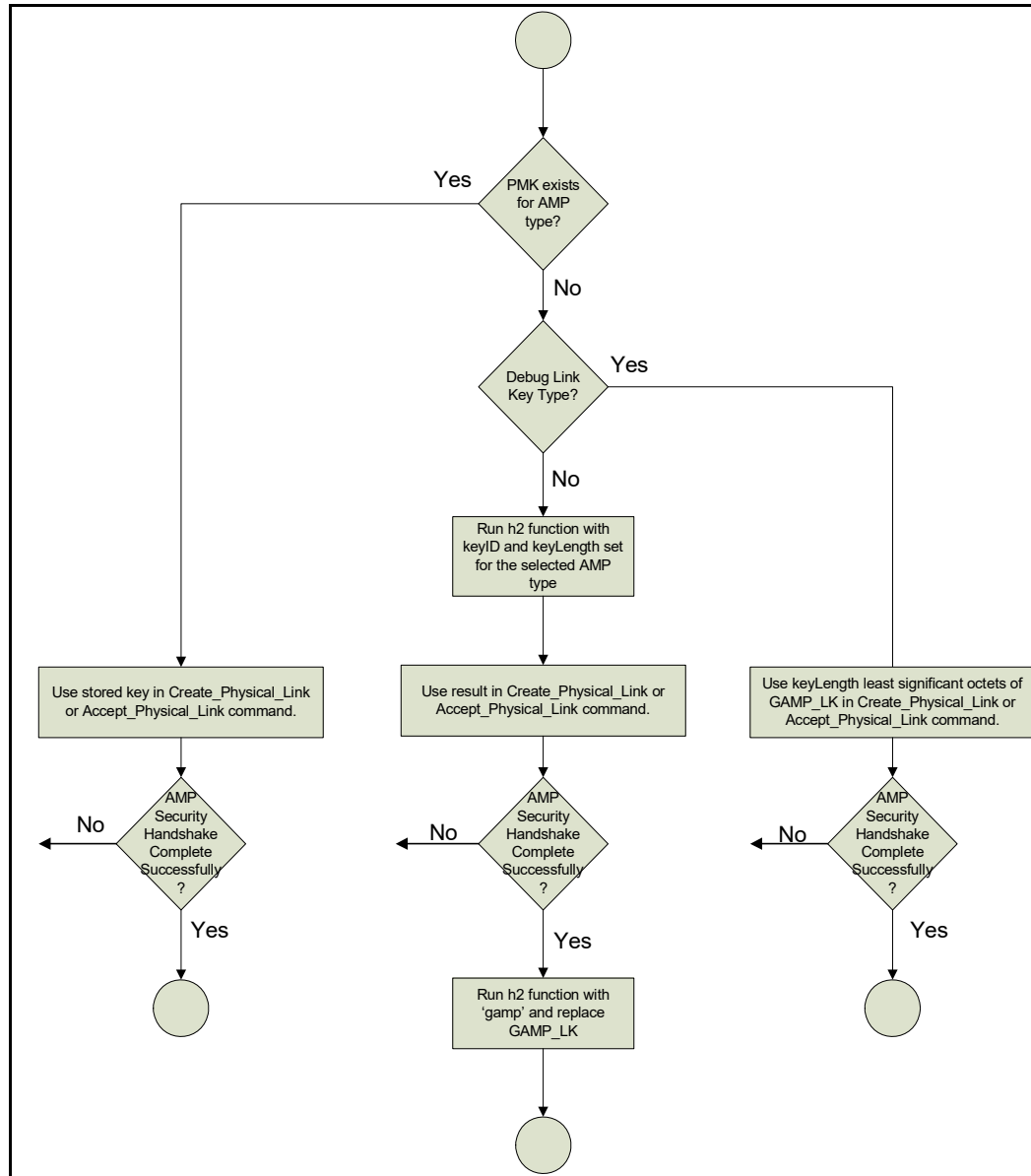


**Figure 8.1:** AMP key generation in Host

Creation of the dedicated AMP keys and updates to the Generic AMP Link Key are the responsibility of the AMP Manager (see [Vol. 3], Part E Section 2.3.3).

### 8.3 DEBUG CONSIDERATIONS

Since the AMP architecture requires encryption to be enabled on each AMP, in order to support sniffers for debug as well as to enable testing over the air, pro-



visions need to be made in the AMP architecture to support debug combination keys. In this document, this is supported in two ways:

- Passing the link key type along with the link key (and length) in the HCI\_Create\_Physical\_Link and HCI\_Accept\_Physical\_Link commands and
- Requiring that the GAMP\_LK is used as the dedicated AMP key when the key type is set to "debug combination key."



## 5 SECURITY ASPECTS

|                                                                                                                                  | Procedure       | Ref. | Support |
|----------------------------------------------------------------------------------------------------------------------------------|-----------------|------|---------|
| 1                                                                                                                                | Authentication  | 5.1  | M       |
| 2                                                                                                                                | Security modes  | 5.2  |         |
|                                                                                                                                  | Security mode 1 |      | E       |
|                                                                                                                                  | Security mode 2 |      | O.1     |
|                                                                                                                                  | Security mode 3 |      | E       |
|                                                                                                                                  | Security mode 4 |      | M       |
| O.1: Security Mode 2 may only be used for backwards compatibility when the remote device does not support Secure Simple Pairing. |                 |      |         |

*Table 5.1: Conformance requirements related to the generic authentication procedure and the security modes defined in this section*

### 5.1 AUTHENTICATION

#### 5.1.1 Purpose

The generic authentication procedure describes how the LMP-authentication and LMP-pairing procedures are used when authentication is initiated by one Bluetooth device towards another, depending on if a link key exists or not and if pairing is allowed or not.

#### 5.1.2 Term on UI level

‘Bluetooth authentication’.

### 5.1.3 Procedure

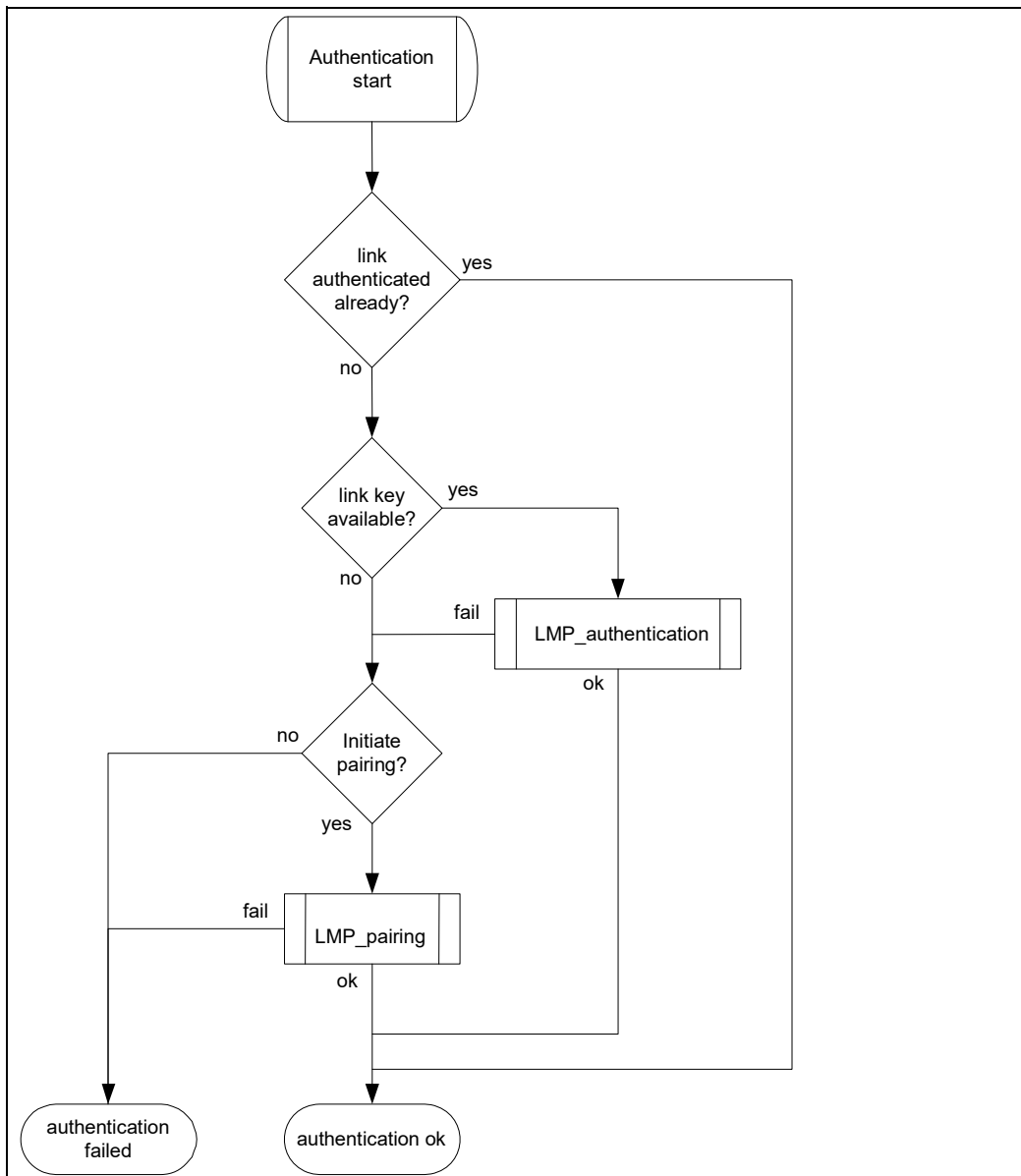


Figure 5.1: Definition of the generic authentication procedure.

### 5.1.4 Conditions

The local device shall initiate authentication after link establishment. The remote device may initiate security during or after link establishment.

## 5.2 SECURITY MODES

The following flow chart provides an overview of the channel establishment procedures.

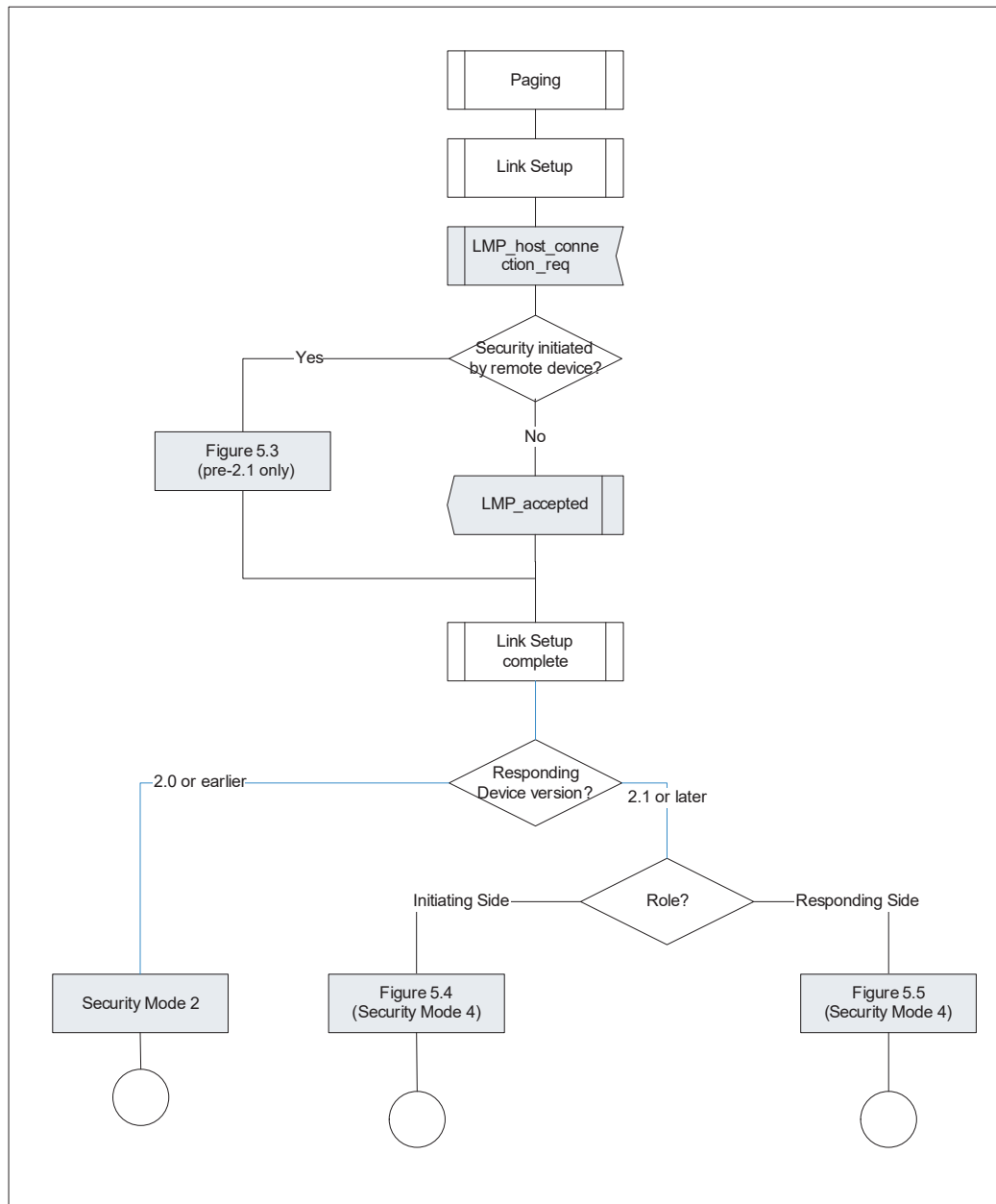


Figure 5.2: Channel establishment with security.

A device may support two security modes simultaneously: security mode 2 for backwards compatibility with remote devices that do not support Secure Simple Pairing and security mode 4 for devices that support Secure Simple Pairing.

### 5.2.1 Legacy Security Modes

Legacy security modes apply to those devices with a Controller or a Host that does not support SSP.



#### 5.2.1.1 Security mode 1 (non-secure)

When a remote Bluetooth device is in security mode 1 it will never initiate any security procedure (i.e., it will never send LMP\_au\_rand, LMP\_in\_rand or LMP\_encryption\_mode\_req).

#### 5.2.1.2 Security mode 2 (service level enforced security)

When a remote Bluetooth device is in security mode 2 it will not initiate any security procedure before a channel establishment request (L2CAP\_ConnectReq) has been received or a channel establishment procedure has been initiated by itself. (The behavior of a device in security mode 2 is further described in [6].) Whether a security procedure is initiated or not depends on the security requirements of the requested channel or service.

A Bluetooth device in security mode 2 should classify the security requirements of its services using at least the following attributes:

- Authorization required
- Authentication required
- Encryption required

*Note: Security mode 1 can be considered (at least from a remote device point of view) as a special case of security mode 2 where no service has registered any security requirements.*

#### 5.2.1.3 Security mode 3 (link level enforced security)

When a remote Bluetooth device is in security mode 3 it will initiate security procedures before it sends LMP\_setup\_complete.

A Bluetooth device in security mode 3 may reject the host connection request (respond with LMP\_not\_accepted to the LMP\_host\_connection\_req) based on settings in the host (e.g., only communication with pre-paired devices allowed).

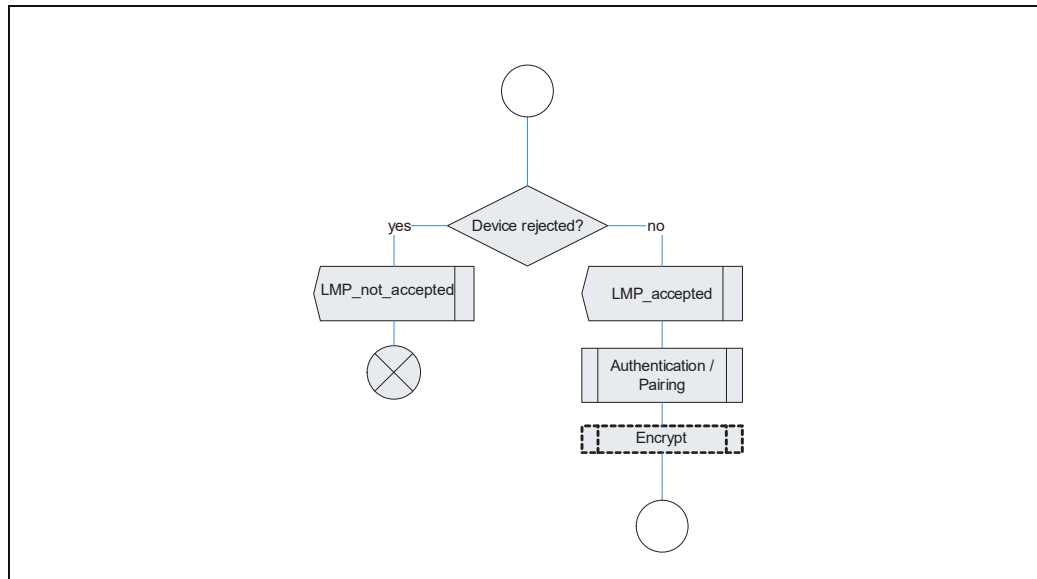


Figure 5.3: Security mode 3 with a legacy remote device

### 5.2.2 Security Mode 4 (service level enforced security)

A Bluetooth device in security mode 4 shall classify the security requirements of its services using at least the following attributes (in order of decreasing security):

- Authenticated link key required
- Unauthenticated link key required
- Security optional – SDP only. Limited to specific services (see [Section 5.2.2.8](#)).

An authenticated link key is a link key where either the numeric comparison, out-of-band, or passkey entry simple pairing association models were used. An authenticated link key has protection against man-in-the-middle (MITM) attacks. To ensure that an authenticated link key is created during the Simple Pairing procedure, the `Authentication_Requirements` parameter should be set to one of the MITM Protection Required options. An *unauthenticated link key* is a link key where the just works Secure Simple Pairing association model was used. An unauthenticated link key does not have protection against MITM attacks.

When both devices support Secure Simple Pairing, GAP shall require at least an unauthenticated link key and enabling encryption for all connections except those allowed to have security level 0 (see [Section 5.2.2.8](#)). A profile or protocol may define services that require more security (e.g., an authenticated link key) or no security (although unencrypted connections are only allowed when connecting to a service allowed to have security level 0). To allow an unauthenticated link key to be created during the Simple Pairing procedure, the



Authentication\_Requirements parameter may be set to one of the MITM Protection Not Required options.

When the device is in Bondable Mode, it shall enable Secure Simple Pairing mode prior to entering Connectable Mode or establishing a link.

A Bluetooth device in security mode 4 shall respond to authentication requests during link establishment when the remote device is in security mode 3 for backwards compatibility reasons.

A Bluetooth device in security mode 4 enforces its security requirements before it attempts to access services offered by a remote device and before it grants access to services it offers to remote devices. Service access may occur via L2CAP channels or via channels established by protocols above L2CAP such as RFCOMM.

For services transmitting unicast data over the connectionless L2CAP channel, the transmitting device shall enforce its security requirements prior to sending data. There is no mechanism for a device receiving data via the L2CAP connectionless channel to prevent unencrypted data from being received. Hence, [Section 5.2.2.1](#) addresses unicast connectionless data transmission together with devices initiating connection-oriented channels while [Section 5.2.2.2](#) covers only devices responding to requests for connection-oriented channel establishment but does not cover unicast connectionless data reception.

#### 5.2.2.1 Security for Access to Remote Service (Initiating Side)

When the responding device does not support Secure Simple Pairing, it may disconnect the link while the initiating device is requesting the PIN to be entered by the user. This may occur due to the lack of an L2CAP channel being present for longer than an implementation-specific amount of time (e.g., a few seconds). When this occurs, the initiating device shall allow the user to complete entering the PIN and shall then re-page the remote device and restart the pairing procedure (see [\[Vol. 2, Part C\] Section 4.2.2 on page 249](#)) using the PIN entered.

##### 5.2.2.1.1 Pairing Required for Access to Remote Service

When a Bluetooth device in security mode 4 initiates access to a remote service via a connection-oriented L2CAP channel and a sufficient link-key is not available, the local device shall perform pairing procedures and enable encryption before sending a channel establishment request (L2CAP\_ConnectReq or a higher-layer channel establishment request such as that of RFCOMM).

When a Bluetooth device in security mode 4 transmits data to a remote service via the unicast connectionless L2CAP channel and a sufficient link-key is not available, the local device shall perform pairing procedures and enable encryption before transmitting unicast data on the connectionless L2CAP channel.



See [Section 5.2.2.8 on page 310](#) for details on determining whether or not a link key is sufficient.

If pairing does not succeed, the local device shall not send a channel establishment request. The local device may re-try pairing up to three (3) times. If pairing fails three consecutive times, the local device shall disconnect the ACL link with error code 0x05 - Authentication Failure.

If the link key generated is not at least as good as the expected or required type, the local device shall fail the establishment and may disconnect the ACL link with error code 0x05 - Authentication Failure.

#### 5.2.2.1.2 Authentication Required for Access to Remote Service

When a Bluetooth device in security mode 4 initiates access to a remote service via a connection-oriented L2CAP channel and a sufficient link key is available for the remote device, it shall authenticate the remote device and enable encryption before sending a channel establishment request (L2CAP\_ConnectReq or higher a layer-channel establishment request such as that of RFCOMM).

When a Bluetooth device in security mode 4 transmits unicast data to a remote service via the connectionless L2CAP channel and security is required for the application and a sufficient link-key is available then the local device shall authenticate the remote device and enable encryption before transmitting unicast data on the L2CAP connectionless channel.

See [Section 5.2.2.8 on page 310](#) for details on determining whether or not a link key is sufficient.

If authentication is required by the service but does not succeed, or if a sufficient link-key is not available, then the local device shall not enable encryption and shall not send a channel establishment request and shall not send any unicast data via the L2CAP connectionless channel for that application. The host may then notify the user and offer to perform pairing.

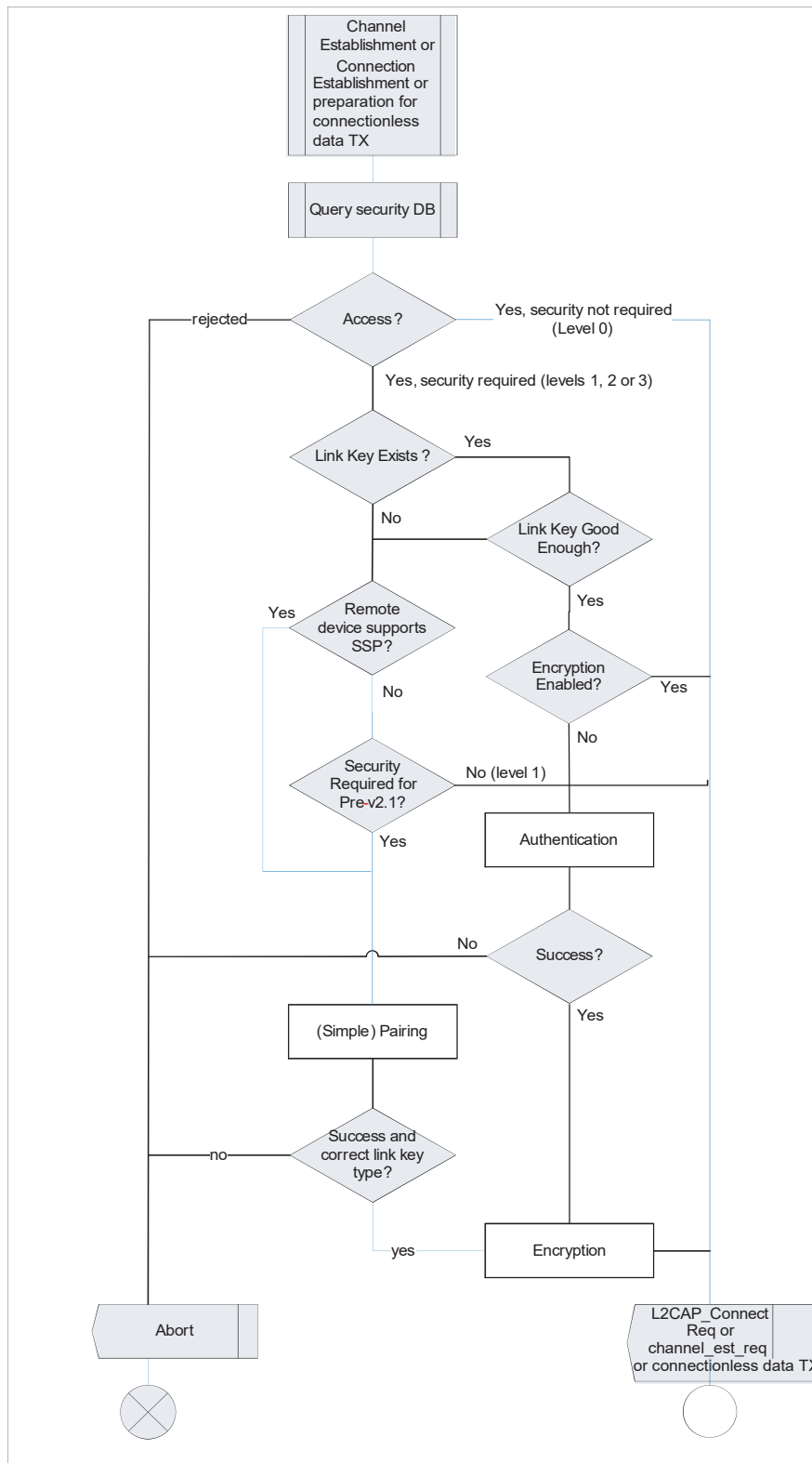


Figure 5.4: Channel establishment using security mode 4 for initiating side



### 5.2.2.2 Security for Access to Local Service by Remote Device (Responding Side)

When a remote device attempts to access a service offered by a Bluetooth device that is in security mode 4 and a sufficient link key exists and authentication has not been performed the local device shall authenticate the remote device and enable encryption after the channel establishment request is received but before a channel establishment confirmation (L2CAP\_ConnectRsp with result code of 0x0000 or a higher-level channel establishment confirmation such as that of RFCOMM) is sent.

When L2CAP is the channel establishment protocol being used for the requested service, an L2CAP\_ConnectRsp signaling packet shall be sent by the responding device containing the result code 0x0001 (connection pending) following receipt of an L2CAP\_ConnectReq and prior to initiating security procedures which can result in prompting the local user for input (e.g., pairing using a PIN or Secure Simple Pairing using either the Passkey entry or Numerical Comparison association models). This will stop the L2CAP RTX timer on the remote device (which may be as short as 1 second) and will invoke the ERTX timer on the remote device, which is a minimum duration of 60 seconds.

See [\[Vol. 3, Part A\] Section 6.1.7 on page 110](#), for additional information on L2CAP RTX and ERTX timers. See also [\[Vol. 3, Part A\] Section 4.3 on page 59](#) for additional information on the L2CAP\_ConnectRsp signalling packet, and the defined result codes.

Higher layer channel establishment protocols should be designed to restrict timeouts to be 30 seconds or longer to allow for user input, or provide mechanisms to dynamically extend timeouts when user input may be required.

If authentication or pairing fails when a remote device is attempting to access a local service, the local device shall send a negative response to the channel establishment request (L2CAP\_ConnectReq or channel\_est\_req) indicating a security issue if possible. If the channel establishment protocol is L2CAP, then the result code 0x0003 (connection refused - security block) shall be sent in the L2CAP\_ConnectRsp signal.

If the remote device has indicated support for Secure Simple Pairing, a channel establishment request is received for a service other than SDP, and encryption has not yet been enabled, then the local device shall disconnect the ACL link with error code 0x05 - Authentication Failure.

#### 5.2.2.2.1 Pairing Required for Access to Local Service by Remote Device

When a remote device attempts to access a service offered by a Bluetooth device that is in security mode 4 and pairing is required due to the link key being absent or insufficient, the local device shall perform pairing procedures and enable encryption after the channel establishment request is received and before a channel establishment confirmation (L2CAP\_ConnectRsp with result



code of 0x0000 or a higher-level channel establishment response such as that of RFCOMM) is sent.

See [Section 5.2.2.6](#) for details on determining whether or not a link key is sufficient.

If pairing does not succeed, then the local device shall not send a channel establishment confirmation. The local device may retry pairing up to three (3) times. If pairing fails three consecutive times, then the local device shall disconnect the ACL link with error code 0x05 - Authentication Failure.

If the link-key generated is not at least as good as the expected or required type, then the local device shall fail the channel establishment and may disconnect the ACL link with error code 0x05 - Authentication Failure.

#### 5.2.2.2.2 Authentication Required for Access to Local Service by Remote Device

See [Section 5.2.2.6](#) for details on determining whether or not a link key is sufficient.

If authentication does not succeed, then the local device shall not send a channel establishment confirmation. The host may at this point notify the user and offer to perform pairing.

A Bluetooth device in security mode 4 shall respond to authentication and pairing requests during link establishment when the remote device is in security mode 3 for backwards compatibility reasons. However, authentication of the remote device shall be performed after the receipt of the channel establishment request is received, and before the channel establishment response is sent.

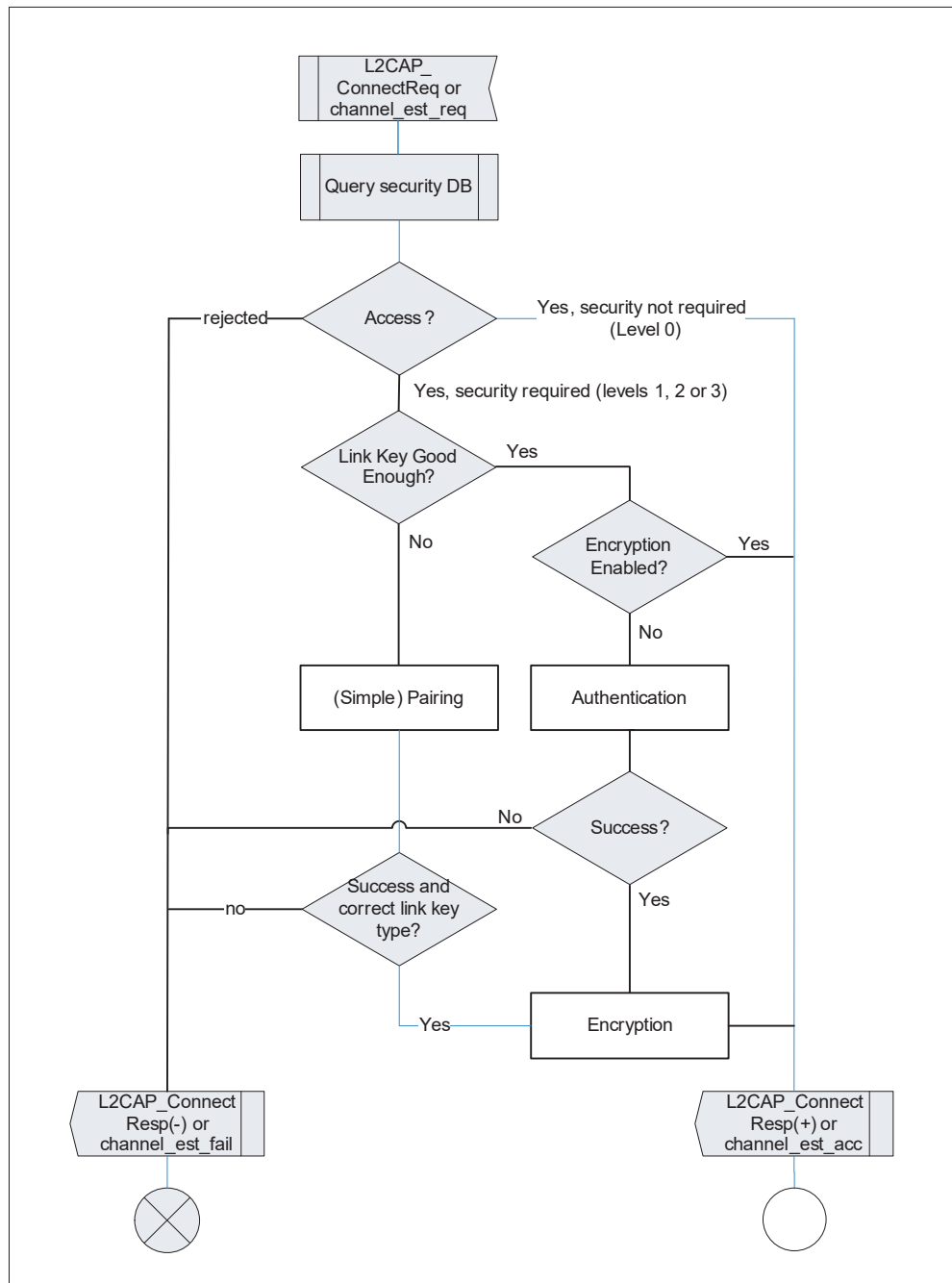


Figure 5.5: Channel establishment using security mode 4 for the responding side

### 5.2.2.3 Simple Pairing after Authentication Failure

When both devices support Secure Simple Pairing all non-SDP connections are encrypted regardless of whether security was required or whether the devices are bonded or not. The initial connection between the two devices will result in a link key through Secure Simple Pairing. Depending on whether or not bonding was performed and the security policy of the initiating device, the



link key may or may not be stored. When the link key is stored, subsequent connections to the same device will use authentication but this may fail if the remote device has deleted the link key. [Table 5.1](#) defines what shall be done depending on the type of the link key and whether bonding was performed or not.

| Link Key Type   | Devices Bonded? | Action to take when Authentication Fails                                                                                                                                                                                                               |
|-----------------|-----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Combination     | No              | Depends on security policy of the device: <ul style="list-style-type: none"> <li>• Option 1: Automatically initiate pairing</li> <li>• Option 2: Notify user and ask if pairing is ok</li> </ul> Option 2 is recommended.                              |
| Combination     | Yes             | Notify user of security failure                                                                                                                                                                                                                        |
| Unauthenticated | No              | Depends on security policy of the device: <ul style="list-style-type: none"> <li>• Option 1: Automatically initiate secure simple pairing</li> <li>• Option 2: Notify user and ask if secure simple pairing is ok.</li> </ul> Option 1 is recommended. |
| Unauthenticated | Yes             | Notify user of security failure                                                                                                                                                                                                                        |
| Authenticated   | No              | Depends on security policy of the device: <ul style="list-style-type: none"> <li>• Option 1: Automatically initiate secure simple pairing</li> <li>• Option 2: Notify user and ask if secure simple pairing is ok</li> </ul> Option 2 is recommended.  |
| Authenticated   | Yes             | Notify user of security failure                                                                                                                                                                                                                        |

*Table 5.1: Simple Pairing after Authentication Failure*

Note that non-bonded authenticated or unauthenticated link keys may be considered disposable by either device and may be deleted at any time.

#### 5.2.2.4 IO Capabilities

Once a connection is established, if the Host determines that security is necessary and both devices support Secure Simple Pairing, the devices perform an IO capability exchange. The purpose of the IO capability exchange is to determine the authentication algorithm used in the Authentication Stage 1 phase of Simple Pairing.

The input and output capabilities are described in [Table 5.2](#):

| Capability | Description                                                |
|------------|------------------------------------------------------------|
| No input   | Device does not have the ability to indicate 'yes' or 'no' |

*Table 5.2: User Input Capabilities*



| Capability | Description                                                                                                                                                                                                                                                                    |
|------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Yes / No   | Device has at least two buttons that are mapped easily to 'yes' and 'no' or the device has a mechanism whereby the user can indicate either 'yes' or 'no' (see note below).                                                                                                    |
| Keyboard   | Device has a numeric keyboard that can input the numbers '0' through '9' and a confirmation. Device also has two buttons that can be easily mapped to 'yes' and 'no' or the device has a mechanism whereby the user can indicate either 'yes' or 'no' (see <b>Note</b> below). |

Table 5.2: User Input Capabilities

**Note:** 'yes' could be indicated by pressing a button within a certain time limit otherwise 'no' would be assumed.

| Capability     | Description                                                                         |
|----------------|-------------------------------------------------------------------------------------|
| No output      | Device does not have the ability to display or communicate a 6 digit decimal number |
| Numeric output | Device has the ability to display or communicate a 6 digit decimal number           |

Table 5.3: User Output Capabilities

#### 5.2.2.5 Mapping of Input / Output Capabilities to IO Capability

The individual input and output capabilities are mapped to a single IO capability which is later used to determine which authentication algorithm will be used.

| Local Output Capacity<br>Local Input Capacity |                 |                |
|-----------------------------------------------|-----------------|----------------|
|                                               | No Output       | Numeric Output |
| No input                                      | NoInputNoOutput | DisplayOnly    |
| Yes / No                                      | NoInputNoOutput | DisplayYesNo   |
| Keyboard                                      | KeyboardOnly    | DisplayYesNo   |

Table 5.4: IO Capability mapping

When a device has OOB authentication information from the remote device, it will indicate it in the LMP\_IO\_Capability\_Res PDU. When either device has OOB information, the OOB authentication method will be used.

The Host may allow the Link Manager to ignore the IO capabilities and use the Numeric Comparison protocol with automatic accept by setting the Authentication\_Requirements parameter to one of the MITM Protection *Not Required* options.



### 5.2.2.6 IO and OOB Capability Mapping to Authentication Stage 1 Method

Determining which association model to use in Authentication Stage 1 is performed in three steps. First, the devices look at the OOB Authentication Data Present parameter received in the remote IO capabilities. If either device has received OOB authentication data then the OOB association model is used. The event of receiving the OOB information is indicated by a device to its peer in the IO Capability Exchange step of simple pairing.

| <div>Device A</div> <div>Device B</div>         | Has not received remote OOB authentication data   | Has received remote OOB authentication table           |
|-------------------------------------------------|---------------------------------------------------|--------------------------------------------------------|
| Has not received remote OOB authentication data | Use the IO capability mapping table               | Use OOB association with<br>ra = 0<br>rb from OOB      |
| Has received remote OOB authentication data     | Use OOB association with<br>ra from OOB<br>rb = 0 | Use OOB association with<br>ra from OOB<br>rb from OOB |

Table 5.5: IO and OOB capability mapping

Second, if neither device has received OOB authentication data and if both devices have set the Authentication\_Requirements parameter to one of the MITM Protection Not Required options, authentication stage 1 shall function as if both devices set their IO capabilities to DisplayOnly (e.g., Numeric comparison with automatic confirmation on both devices).

Finally, if neither device has received OOB authentication data and if one or both devices have set the Authentication\_Requirements parameter to one of the *MITM Protection Required* options, the IO and OOB capabilities are mapped to the authentication stage 1 method as defined in the following table. A Host that has set the Authentication\_Requirements parameter to one of the *MITM Protection Required* options shall verify that the resulting Link Key is an Authenticated Combination Key (see [Section 7.7.24 in Host Controller Interface Functional Specification](#)). The table also lists whether the combination key results in an authenticated or unauthenticated link key.

Note: The "DisplayOnly" IO capability only provides unidirectional authentication.

### 5.2.2.7 Out of Band (OOB)

An out of band mechanism may also be used to communicate discovery information as well as other information related to the pairing process.

The contents of the OOB data block are:

Mandatory contents:



| <b>Initiator A</b><br><b>B Responder</b> | <b>Display Only</b>                                                                     | <b>DisplayYesNo</b>                                                                     | <b>KeyboardOnly</b>                                                                    | <b>NoInputNoOutput</b>                                                                  |
|------------------------------------------|-----------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------|
| <b>DisplayOnly</b>                       | Numeric Comparison with automatic confirmation on both devices.<br><br>Unauthenticated  | Numeric Comparison with automatic confirmation on device B only.<br><br>Unauthenticated | Passkey Entry: Responder Display, Initiator Input.<br><br>Authenticated                | Numeric Comparison with automatic confirmation on both devices.<br><br>Unauthenticated  |
| <b>DisplayYesNo</b>                      | Numeric Comparison with automatic confirmation on device A only.<br><br>Unauthenticated | Numeric Comparison: Both Display, Both Confirm.<br><br>Authenticated                    | Passkey Entry: Responder Display, Initiator Input.<br><br>Authenticated                | Numeric Comparison with automatic confirmation on device A only.<br><br>Unauthenticated |
| <b>Keyboard Only</b>                     | Passkey Entry: Initiator Display, Responder Input.<br><br>Authenticated                 | Passkey Entry: Initiator Display, Responder Input.<br><br>Authenticated                 | Passkey Entry: Initiator and Responder Input.<br><br>Authenticated                     | Numeric Comparison with automatic confirmation on both devices.<br><br>Unauthenticated  |
| <b>NoInputNoOutput</b>                   | Numeric Comparison with automatic confirmation on both devices.<br><br>Unauthenticated  | Numeric Comparison with automatic confirmation on device B only.<br><br>Unauthenticated | Numeric Comparison with automatic confirmation on both devices.<br><br>Unauthenticated | Numeric Comparison with automatic confirmation on both devices.<br><br>Unauthenticated  |

Table 5.6: IO Capability Mapping to Authentication Stage 1

- Length (2 bytes)
- BD\_ADDR (6 bytes)

Optional contents:

- Class of Device (3 bytes)
- Simple Pairing Hash C (16 bytes)
- Simple Pairing Randomizer R (16 bytes)
- Local name (variable length)
- Other information



The length field includes all bytes in the OOB data block including the length field itself. The BD\_ADDR will be a fixed field in the beginning of the OOB data block. Following the BD\_ADDR will be zero or more EIR tag fields containing optional contents. The EIR tag format will be used for the optional contents.

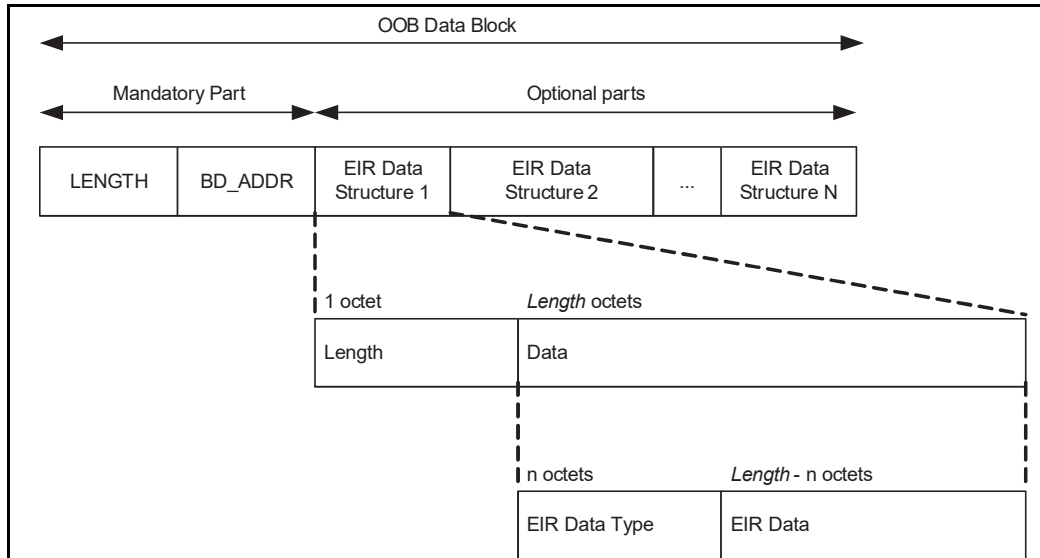


Figure 5.6: OOB Data Block Format

If Simple Pairing fails when one or both devices have OOB Authentication Data present, both devices shall discard the OOB Authentication Data and the device that originally initiated authentication shall re-initiate authentication. Note that although the user may get involved in authentication as defined by the IO capabilities of the two devices, falling back to the in-band authentication method will prevent deadlock conditions when one or both devices has stale OOB Authentication Data.

#### 5.2.2.8 Security Database

A Bluetooth device in security mode 4 shall classify the security requirements of its services using at least the following levels attributes (in order of decreasing security) for use when pairing with remote devices supporting Secure Simple Pairing:

- Level 3, for services with the following attributes:
  - MITM protection required
  - Encryption required
  - User interaction acceptable
- Level 2, for services with the following attributes:
  - MITM protection not required
  - Encryption required
- Level 1, for services with the following attributes:



MITM protection not required

Minimal user interaction desired

One additional level is permitted for L2CAP fixed signaling channels (CIDs 0x0001, 0x0003 and 0x003F), SDP and for some service traffic sent via the connectionless L2CAP channel for services that do not require security but shall not be used for any other service traffic:

- Level 0: Service requires the following:

MITM protection not required

No encryption required

No user interaction required

Level 0 may only be used for Services with the following Service Class UUIDs:

0x1000 + BASE\_ID (Service Discovery Server)

The security level required for each service offered should be stored in a security database that is accessed to determine the type of link key that is required for access to the respective service. The security level required for service data transmitted on an L2CAP connection-oriented channel may differ from the security level required for service data transmitted on another L2CAP connection-oriented channel or on the connectionless L2CAP channel. [Table 5.7 on page 311](#) shows the type of link key required for each security level for both remote devices that support Secure Simple Pairing (v2.1 + EDR remote devices) and for those that do not (pre-v2.1 + EDR remote devices).

| Security Level Required for Service                                                                                                                             | Link Key type required for remote devices | Link Key type required for pre-v2.1 remote devices | Comments        |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------|----------------------------------------------------|-----------------|
| Level 3 <ul style="list-style-type: none"> <li>• Strong MITM protection desired</li> <li>• Encryption desired</li> <li>• User interaction acceptable</li> </ul> | Authenticated                             | Combination (16 digit PIN recommended)             | High Security   |
| Level 2 <ul style="list-style-type: none"> <li>• MITM protection not necessary</li> <li>• Encryption desired</li> </ul>                                         | Unauthenticated                           | Combination                                        | Medium Security |

Table 5.7: Security Level mapping to link key requirements



| Security Level Required for Service                                                                                                                                                   | Link Key type required for remote devices | Link Key type required for pre-v2.1 remote devices | Comments                                                                                                                                                                                                        |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------|----------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Level 1 <ul style="list-style-type: none"> <li>• MITM protection not necessary</li> <li>• Encryption not necessary<sup>1</sup></li> <li>• Minimal user interaction desired</li> </ul> | Unauthenticated                           | None                                               | Low Security                                                                                                                                                                                                    |
| Level 0 <ul style="list-style-type: none"> <li>• MITM protection not necessary</li> <li>• Encryption not necessary</li> <li>• No user interaction desired</li> </ul>                  | None                                      | None                                               | Permitted only for SDP and service data sent via either L2CAP fixed signaling channels or the L2CAP connectionless channel to PSMs that correspond to service class UUIDs which are allowed to utilize Level 0. |

Table 5.7: Security Level mapping to link key requirements

1. Though encryption is not necessary for the service for Level 1, this specification mandates the use of encryption when the remote device is v2.1+EDR for all services other than SDP.

An *authenticated* link key is a link key where either the numeric comparison, out-of-band, or passkey entry simple pairing association models were used. An authenticated link key has protection against MITM attacks. To ensure that an authenticated link key is created during the Simple Pairing procedure, the Authentication\_Requirements parameter should be set to one of the *MITM Protection Required* options.

An *unauthenticated* link key is a link key where the “Just Works” simple pairing association model was used (see [Vol. 1, Part A] Section 5.1.4 on page 87). An unauthenticated link key does not have protection against MITM attacks. To allow an unauthenticated link key to be created during the Simple Pairing procedure, the Authentication\_Requirements parameter may be set to one of the *MITM Protection Not Required* options.

A combination link key is a link key where the v2.0 pairing mechanism was used to generate the link-key (see [Vol. 2, Part C] Section 4.2.2.4 on page 251).

When both devices support Secure Simple Pairing, GAP shall require at least an unauthenticated link key and enable encryption for service traffic sent or received via connection-oriented L2CAP channels. A profile or protocol may define services that require more security (for example, an authenticated link



key) or no security in the case of SDP or service traffic sent via the L2CAP connectionless channel for services that do not require security.

When the device is in Bondable Mode, it shall enable Secure Simple Pairing mode prior to entering Connectable Mode or establishing a link.

A Bluetooth device in security mode 4 shall respond to authentication and pairing requests during link establishment when the remote device is in security mode 3 for backwards compatibility reasons. See [Section 5.2.1.3](#) for more information.

The remote Controller's and remote Host's support for Secure Simple Pairing shall be determined by the Link Manager Secure Simple Pairing (Host Support) feature bit.

A previously generated link key is considered "sufficient" if the link key type is of the type required for the service, or of a higher strength. Authenticated link keys are considered higher strength than Unauthenticated or Combination keys. Unauthenticated link keys are considered higher strength than Combination keys.

## 10 LE SECURITY ASPECTS

This section defines the modes and procedures that relate to the security of a connection. The following modes and procedures are defined:

- LE security mode 1
- LE security mode 2
- Authentication procedure
- Authorization procedure
- Connection data signing procedure
- Authenticate signed data procedure

Requirements for a device to support the LE security modes and procedures is shown in [Table 10.1](#).

### 10.1 REQUIREMENTS

| Security Modes and Procedures      | Ref.                   | Broadcaster | Observer | Peripheral | Central |
|------------------------------------|------------------------|-------------|----------|------------|---------|
| LE Security mode 1                 | <a href="#">10.2.1</a> | E           | E        | O          | O       |
| LE Security mode 2                 | <a href="#">10.2.2</a> | E           | E        | O          | O       |
| Authentication procedure           | <a href="#">10.3</a>   | E           | E        | O          | O       |
| Authorization procedure            | <a href="#">10.5</a>   | E           | E        | O          | O       |
| Connection data signing procedure  | <a href="#">10.4.1</a> | E           | E        | O          | O       |
| Authenticate signed data procedure | <a href="#">10.4.2</a> | E           | E        | O          | O       |

Table 10.1: Requirements related to security modes and procedures

### 10.2 LE SECURITY MODES

The security requirements of a device, a service or a service request are expressed in terms of a security mode and security level. Each service or service request may have its own security requirement. The device may also have a security requirement. A physical connection between two devices shall operate in only one security mode.

There are two LE security modes, LE security mode 1 and LE security mode 2.

#### 10.2.1 LE Security Mode 1

LE security mode 1 has three security levels:



1. No security (No authentication and no encryption)
2. Unauthenticated pairing with encryption
3. Authenticated pairing with encryption

A connection operating in LE security mode 1 level 2 shall also satisfy the security requirements for LE security mode 1 level 1.

A connection operating in LE security mode 1 level 3 shall also satisfy the security requirements for LE security mode 1 level 2 or LE security mode 1 level 1.

A connection operating in LE security mode 1 level 3 shall also satisfy the security requirements for LE security mode 2.

### 10.2.2 LE Security Mode 2

LE security mode 2 has two security levels:

1. Unauthenticated pairing with data signing
2. Authenticated pairing with data signing

LE security mode 2 shall only be used for connection based data signing.

Data signing as defined in [Section 10.4](#) shall not be used when a connection is operating in LE security mode 1 level 2 or LE security mode 1 level 3.

### 10.2.3 Mixed Security Modes Requirements

If there are requirements for both LE security mode 1 and LE security mode 2 level 2 for a given physical link then LE security mode 1 level 3 shall be used.

If there are requirements for both LE security mode 1 level 3 and LE security mode 2 for a given physical link then LE security mode 1 level 3 shall be used.

If there are requirements for both LE security mode 1 level 2 and LE security mode 2 level 1 for a given physical link then LE security mode 1 level 2 shall be used.

## 10.3 AUTHENTICATION PROCEDURE

The authentication procedure describes how the required security is established when a device initiates a service request to a remote device and when a device receives a service request from a remote device. The authentication procedure covers both LE security mode 1 and LE security mode 2. The authentication procedure shall only be initiated after a connection has been established.



There are two types of pairing: authenticated pairing or unauthenticated pairing. Authenticated pairing involves performing the pairing procedure defined in [\[Vol. 3\], Part H Section 2.1](#) with the authentication set to 'MITM protection'. Unauthenticated pairing involves performing the pairing procedure with authentication set to 'No MITM protection'.

### 10.3.1 Receiving a Service Request

A service request is a request to perform a GATT procedure. A service request may also be a request to perform a procedure using other higher layer protocols on top of L2CAP.

When a device receives a service request from a remote device it shall behave as follows:

- The Server's security database specifies the security settings required to accept a service request from a remote device. If no security is required, the service request shall be accepted; otherwise pairing is required before the service request can be accepted.
- If pairing has not occurred or an authenticated pairing is required but only an unauthenticated pairing has occurred, the service request shall be rejected with the error code "Insufficient Authentication".
- If the required pairing has occurred and encryption is required (LE security mode 1) but encryption is not enabled, the service request shall be rejected with the error code "Insufficient Encryption". If the encryption is enabled with insufficient key size then the service request shall be rejected with the error code "Insufficient Encryption Key Size."
- If authenticated pairing has occurred and encryption is not required (LE security mode 2) then Data Signing shall be used if CSRK has been exchanged, else the Host shall re-pair and exchange CSRK before proceeding with Data Signing.

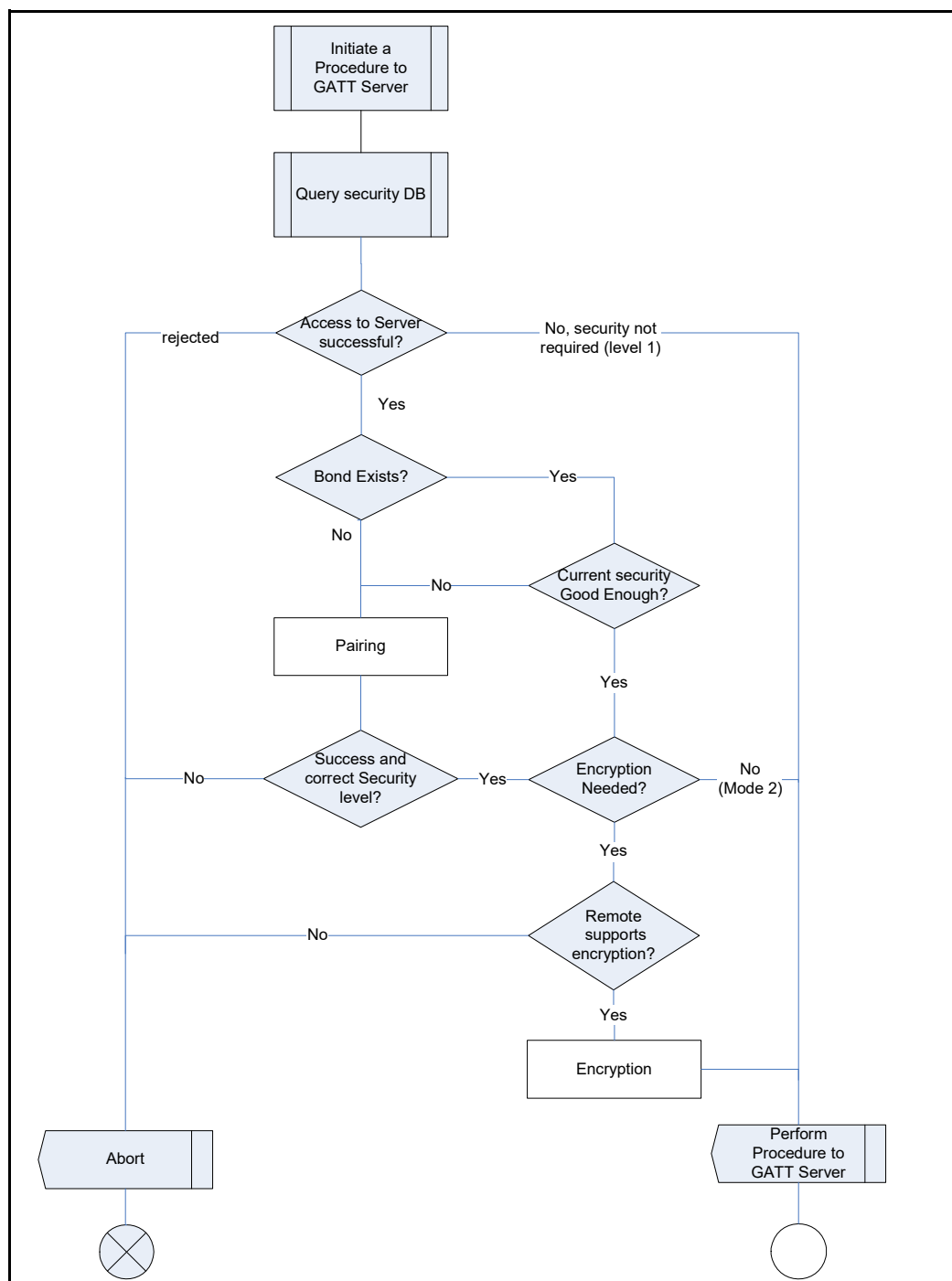


Figure 10.1: Flow chart for a server handling the service request

### 10.3.2 Initiating a Service Request

When a device initiates a service request to a remote device it shall behave as follows:



- The Client's security database specifies the security required to initiate a service request. If no security is required then the service request shall proceed otherwise pairing is required. If security is required but pairing has not occurred in a previous bonding procedure then pairing shall occur during the current connection.
- If pairing has not occurred or an authenticated pairing is required but only an unauthenticated pairing has occurred, the pairing procedure shall be executed with the required authentication settings. If the pairing procedure fails then the service request shall be aborted.
- If authenticated pairing has occurred but the encryption key size is insufficient the pairing procedure shall be executed with the required encryption key size. If the pairing procedure fails then the service request shall be aborted.
- If the required pairing has occurred and encryption is required (LE security mode 1) then encryption shall be enabled before the service request proceeds. If encryption is required but encryption is not enabled and the remote device supports encryption then encryption shall be enabled using the encryption procedure as defined in [Section 10.6](#). Once encryption is enabled the service request shall proceed. If encryption is required but not supported by the remote device then the service request shall be aborted.
- If encryption is not required and authenticated pairing has occurred (LE security mode 2) then the data signing procedure shall be used when making the service request.

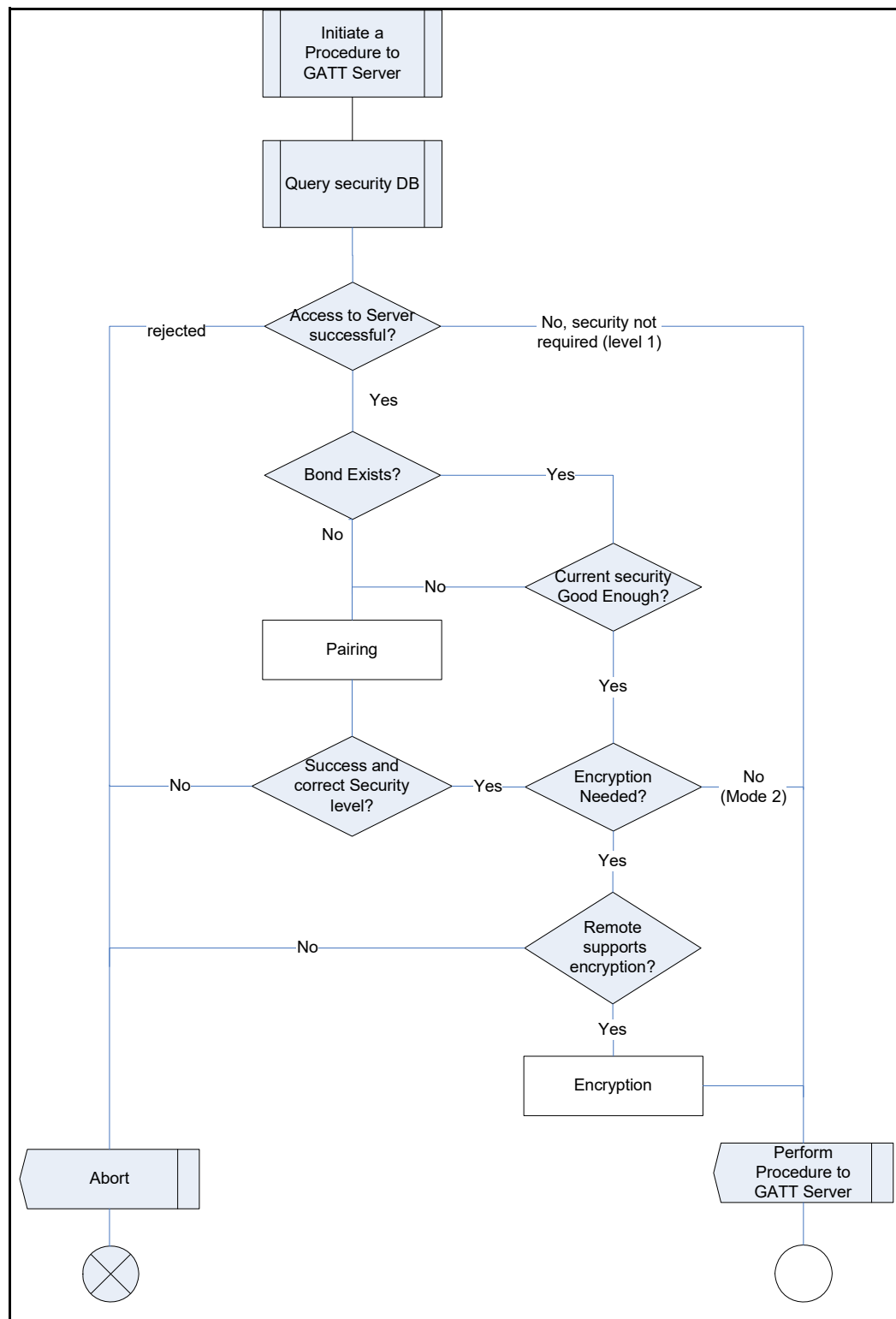


Figure 10.2: Flow chart for a client issuing a service request

## 10.4 DATA SIGNING

The data signing is used for transferring authenticated data between two devices in an unencrypted connection. The data signing method is used by services that require fast connection set up and fast data transfer.

If a service request specifies LE security mode 2, the connection data signing procedure shall be used.

### 10.4.1 Connection Data Signing Procedure

A device shall generate a new Connection Signature Resolving Key CSRK for each peer device to which it sends signed data in a connection. CSRK is defined in [\[Vol. 3\], Part H Section 2.4.2.2](#).

The data shall be formatted using the Signing Algorithm as defined in [\[Vol. 3\], Part H Section 2.4.5](#) where  $m$  is the Data PDU to be signed,  $k$  is the CSRK and the SignCounter is the counter value. A Signature is composed of the counter value and the Message Authentication Code (MAC) generated by the Signing Algorithm. The counter value shall be incremented by one for each new Data PDU sent.

The format of signed data is shown in [Figure 10.3](#).

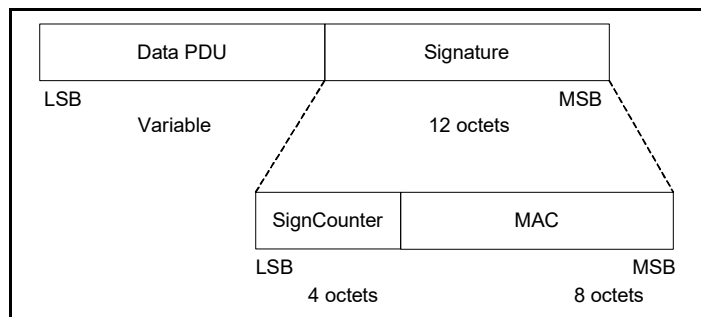


Figure 10.3: Generic format of signed data

### 10.4.2 Authenticate Signed Data Procedure

A device connected to a peer device sending signed data can authenticate the signed data by using the stored CSRK.

The signed data shall be authenticated by performing the Signing Algorithm where  $m$  is the Data PDU to be authenticated,  $k$  is the stored CSRK and the SignCounter is the received counter value. If the MAC computed by the Signing Algorithm does not match the received MAC, the verification fails and the Host shall ignore the received Data PDU

The receiving device shall protect against a replay attack by comparing the received SignCounter with previously received SignCounter from the same



peer device. If the SignCounter was previously used then the receiving device shall ignore the Data PDU and respond with error code “Signature Authentication Failed”, if permitted. The receiving device should protect from repeated failures by implementing a timer similar to the pairing timer as defined in [Vol. 3], Part H Section 2.3.6.

## 10.5 AUTHORIZATION PROCEDURE

A service may require authorization before allowing access. Authorization is a confirmation by the user to continue with the procedure. Authentication does not necessarily provide authorization. Authorization may be granted by user confirmation after successful authentication.

## 10.6 ENCRYPTION PROCEDURE

A Central may encrypt a connection using the Encryption Session Setup as defined in [Vol. 3], Part H Section 2.4.4 to provide integrity and confidentiality.

A Peripheral may encrypt a connection using the Slave Initiated Security Request as defined in [Vol. 3], Part H Section 2.4.6 to provide integrity and confidentiality.

## 10.7 PRIVACY FEATURE

The privacy feature provides a level of privacy which makes it more difficult for an attacker to track a device over a period of time. The requirements for a device to support the privacy feature are defined in Table 10.2.

| Privacy Requirements                                                                                                                                                                                     | Ref.     | Broadcaster | Observer | Peripheral | Central |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------|-------------|----------|------------|---------|
| Privacy feature                                                                                                                                                                                          | 10.7     | O           | O        | O          | O       |
| Non-resolvable private address generation procedure                                                                                                                                                      | 10.8.2.1 | E           | C1       | C2         | C1      |
| Resolvable private address generation procedure                                                                                                                                                          | 10.8.2.2 | C3          | E        | C3         | E       |
| Resolvable private address resolution procedure                                                                                                                                                          | 10.8.2.3 | E           | C2       | E          | C3      |
| C1: Mandatory if privacy feature and Active scanning are supported, else excluded<br>C2: Optional if privacy is supported, else excluded<br>C3: Mandatory if privacy feature is supported, else excluded |          |             |          |            |         |

Table 10.2: Requirements related to privacy feature

### 10.7.1 Privacy Feature in a Peripheral

Support of the privacy feature in a Peripheral depends on the presence and the value of two characteristics: Peripheral Privacy Flag characteristic as defined in [Section 12.3](#) and the Reconnection Address characteristic as defined in [Section 12.4](#). When the Peripheral Privacy Flag characteristic is set to 1, privacy is enabled (also known as privacy enabled Peripheral). [Table 10.3](#) shows privacy support in the Peripheral. A privacy enabled Peripheral that does not expose the Reconnection Address characteristic may use a resolvable private address as the advertiser's device address when sending advertising events, depending on the mode that the Peripheral is operating in. A privacy enabled Peripheral that does expose the Reconnection Address characteristic may use the reconnection address as the advertiser's device address only if it has been set by the Central when sending advertising events depending on the mode that the Peripheral is operating in. When the Peripheral Privacy Flag characteristic is set to 0, privacy is disabled. When privacy is disabled the peripheral shall use a static or public address. When privacy is disabled, the permission of the Reconnection Address characteristic value shall be read-only, and shall not be considered as set by the Central.

| Peripheral Privacy Flag characteristic present and set to 1 (privacy enabled)? | Reconnection Address characteristic present? | Privacy support in Peripheral                                           |
|--------------------------------------------------------------------------------|----------------------------------------------|-------------------------------------------------------------------------|
| No                                                                             | No                                           | No privacy                                                              |
| Yes                                                                            | No                                           | Privacy in undirected connectable mode only                             |
| No                                                                             | Yes                                          | No privacy                                                              |
| Yes                                                                            | Yes                                          | Privacy in directed connectable mode and in undirected connectable mode |

Table 10.3: *privacy feature in a Peripheral*

Note: A privacy enabled Peripheral may write one or more Central's static or public address in the White List of the Controller. A privacy enabled Peripheral may also write one or more reconnection addresses in the White List of the Controller. Static, public and reconnection addresses stored in the White List allow the Controller of the Peripheral to autonomously establish a connection with the Central that initiates a connection with the address that matches the address in the White List.

A Peripheral that supports the Peripheral Privacy Flag characteristic may allow a bonded Central to enable or disable the privacy feature. If the number of bonds on the Peripheral is less than or equal to one, the Peripheral Privacy Flag characteristic value may be writable. If a Peripheral has more than one bond, the permission of the Peripheral Privacy Flag characteristic value shall be read-only.



### 10.7.2 Privacy Feature in a Central

A device in the Central may support the privacy feature by use of the non-resolvable address or the reconnection address as the device address.

If the privacy feature is supported in a Central (also known as privacy enabled Central), the Central shall use a non-resolvable private address during active scanning.

When a Central connects to a privacy enabled Peripheral with the reconnection address characteristic, the Central shall write a new reconnection address in the reconnection address characteristic in the Peripheral. See general connection establishment procedure in [Section 9.3.6](#). The privacy enabled Central may use the reconnection address when initiating a connection to a privacy enabled Peripheral.

When a Central connects with a Peripheral that supports the Peripheral Privacy Flag characteristic the Central may be allowed to enable or disable the privacy feature in the Peripheral.

Note: A privacy enabled Central can write the reconnection address in the White List.

## 10.8 RANDOM DEVICE ADDRESS

Random device addresses are defined in the Link Layer section in [\[Vol. 6\], Part B Section 1.3](#).

For the purposes of this profile, the random device address may be of either of the following two sub-types:

- Static address
- Private address.

The term random device address refers to both static and private address types.

The transmission of a random device address is optional. A device shall accept the reception of a random device address from a remote device.

The private address may be of either of the following two sub-types:

- Non-resolvable private address
- Resolvable private address

### 10.8.1 Static Address

A static address is a 48-bit randomly generated address and shall meet the following requirements:



- The two most significant bits of the static address shall be equal to '1'
- All bits of the random part of the static address shall not be equal to '1'
- All bits of the random part of the static address shall not be equal to '0'

The format of a static address is shown in [Figure 10.4](#).

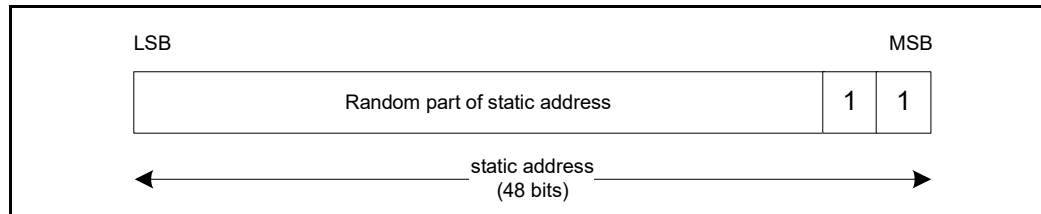


Figure 10.4: Format of static address

A device may choose to initialize its static address to a new value after each power cycle. A device shall not change its static address value once initialized until the device is power cycled.

Note: If the static address of a device is changed then the address stored in peer devices will not be valid and the ability to reconnect using the old address will be lost.

## 10.8.2 Private address

The private address may be of either of the following two sub-types:

- Non-resolvable private address
- Resolvable private address

The generation of a reconnection address shall use the non-resolvable private address generation procedure.

### 10.8.2.1 Non-Resolvable Private Address Generation Procedure

To generate a non-resolvable address or a reconnection address, the Host shall generate a 48-bit address with the following requirements:

- The two most significant bits of the address shall be equal to '0'
- All bits of the random part of the address shall not be equal to '0'
- All bits of the random part of the address shall not be equal to '1'
- The address shall not be equal to the static address
- The address shall not be equal to the public address

The format of a non-resolvable private address is shown in [Table 10.5](#).

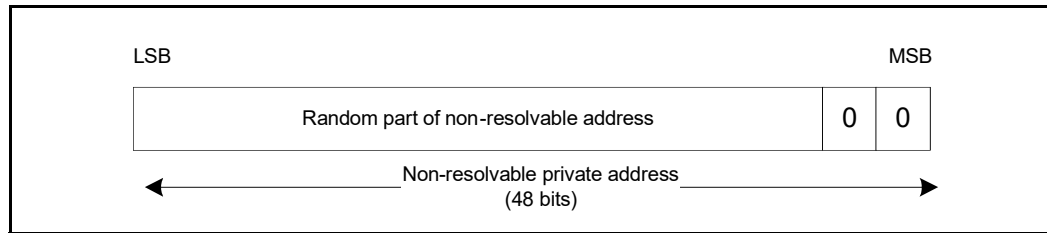


Figure 10.5: Format of non-resolvable private address

### 10.8.2.2 Resolvable Private Address Generation Procedure

To generate a resolvable private address the Host must have its Identity Resolving Key (IRK). The resolvable private address shall be generated with the IRK and a randomly generated 24-bit random number. The random number is known as *prand* and shall meet the following requirements:

- The two most significant bits of *prand* shall be equal to '0' and '1' as shown in Figure 10.6
- All bits of the random field of *prand* shall not be equal to '0'
- All bits of the random field of *prand* shall not be equal to '1'

The format of the resolvable private address is shown in Figure 10.6.

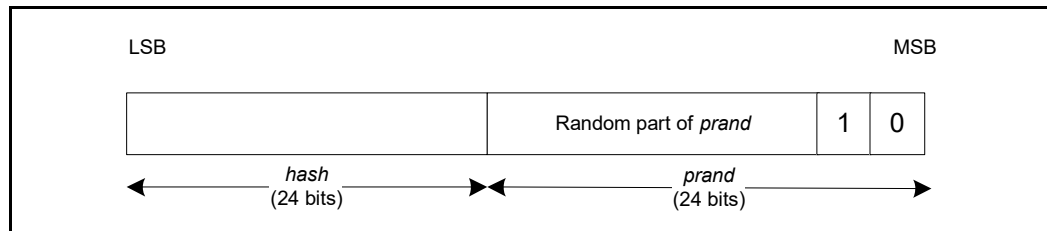


Figure 10.6: Format of resolvable private address

The *hash* is generated using the random address function *ah* defined in [Vol. 3], Part H Section 2.2.2 with the input parameter *k* set to the device's IRK and the input parameter *r* set to *prand*.

$$hash = ah(IRK, prand)$$

The *prand* and *hash* are concatenated to generate the random address (*randomAddress*) in the following manner:

$$randomAddress = hash || prand$$

The least significant octet of *hash* becomes the least significant octet of *randomAddress* and the most significant octet of *prand* becomes the most significant octet of *randomAddress*.



### 10.8.2.3 Resolvable Private Address Resolution Procedure

The Host can resolve a resolvable private address for all peer devices where the Host has the peer device's IRK using this procedure. If a resolvable private address is resolved, the Host can associate this address with the peer device.

The resolvable private address (*rpAddress*) is divided into a 24-bit random part (*prand*) and a 24-bit hash part (*hash*). The least significant octet of the *rpAddress* becomes the least significant octet of *hash* and the most significant octet of *rpAddress* becomes the most significant octet of *prand*.

A *localHash* value is then generated using the random address hash function *ah* defined in [Vol. 3], Part H Section 2.2.2 with the input parameter *k* set to IRK of the known device and the input parameter *r* set to the *prand* value extracted from the *rpAddress*.

$$localHash = ah(IRK, prand)$$

The *localHash* value is then compared with the *hash* value extracted from *rpAddress*. If the *localHash* value matches the extracted *hash* value then the identity of the peer device has been resolved.

If a Host has more than one stored IRK, the Host repeats the above procedure for each stored IRK to determine if the received resolvable private address is associated with a stored IRK, until either address resolution is successful for one of the IRKs or all have been tried.