

Specification of the Bluetooth System

Wireless connections made easy

Master Table of Contents & Compliance Requirements

Covered Core Package version: 1.2
Current Master TOC issued:
05 November 2003





Revision History

The Revision History is shown in the [“Appendix” on page 53\[vol. 0\]](#).

Contributors

The persons who contributed to this specification are listed in [“Appendix” on page 53\[vol. 0\]](#).

Web Site

This specification can also be found on the Bluetooth website:
<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 and to adopt a process for adding new Bluetooth® profiles after the release of the Specification.

Copyright © 1999, 2000, 2001, 2002, 2003

Agere Systems, Inc.,
Ericsson Technology Licensing, AB,
IBM Corporation,
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 & Compliance Requirements
Part A

MASTER TABLE OF CONTENTS

Bluetooth Specification
Including Core v1.2

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.





MASTER TOC FOR THE BLUETOOTH SPECIFICATION

In the following table:

- The TOC for each Volume starts at the top of a page.
- The Volume No. is written in red and followed by the name of the Volume.

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 cover 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 will 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
Master TOC for the Bluetooth Specification 7

Part B
BLUETOOTH COMPLIANCE REQUIREMENTS

Contents 43

1 Scope 45

2 Definitions 47

3 Legal Aspects 49

4 Introduction to the Bluetooth Qualification Program 51



Part C

APPENDIX

Contents	55
1 Revision History.....	57
1.1 [vol 0] Master TOC & Compliance Requirements	57
1.1.1 Bluetooth Compliance Requirements.....	57
1.2 [Vol 1] Architecture & Terminology Overview	57
1.3 [Vol 2 & 3] Core System Package	58
2 Contributors	59
2.1 [vol 0] Master TOC & Compliance Requirements	59
2.1.1 Part B: Bluetooth Compliance Requirements	59
2.2 [vol 1] Architecture & Terminology Overview	59
2.2.1 Part A: Architectural Overview	59
2.2.2 Part B: Acronyms & Abbreviations	60
2.2.3 Part C: Changes from Bluetooth Specification v1.1	60
2.3 [Vol 2] Core System Package, Controller	61
2.3.1 Part A: Radio Specification	61
2.3.2 Part B: Baseband Specification.....	62
2.3.3 Part C: Link Manager Protocol.....	65
2.3.4 Part D: Error Codes.....	66
2.3.5 Part E: Bluetooth Host Controller Interface Functional Specification	67
2.3.6 Part F: Message Sequence Charts	68
2.3.7 Part G: Sample Data	69
2.3.8 Part H: Security Specification.....	69
2.4 [Vol 3] Core System Package, Host.....	70
2.4.1 Part A: Logical Link Control and Adaptation Protocol Specification	70
2.4.2 Part B: Service Discovery Protocol (SDP)	71
2.4.3 Part C Generic Access Profile.....	72
2.4.4 Part D: Test Support.....	72



Specification Volume 1
Architecture & Terminology Overview

Table of Contents	5
--------------------------------	----------

Part A

ARCHITECTURE

Contents	11
1 General Description	13
1.1 Overview of Operation	13
1.2 Nomenclature	15
2 Core System Architecture	19
2.1 Core Architectural Blocks	22
2.1.1 Channel manager	22
2.1.2 L2CAP resource manager	22
2.1.3 Device manager	23
2.1.4 Link manager	23
2.1.5 Baseband resource manager	23
2.1.6 Link controller	24
2.1.7 RF	24
3 Data Transport Architecture	25
3.1 Core Traffic Bearers	26
3.1.1 Framed data traffic	27
3.1.2 Unframed data traffic	28
3.1.3 Reliability of traffic bearers	28
3.2 Transport Architecture Entities	30
3.2.1 Bluetooth generic packet structure	30
3.3 Physical Channels	32
3.3.1 Basic piconet channel	33
3.3.2 Adapted piconet channel	34
3.3.3 Inquiry scan channel	35
3.3.4 Page scan channel	36
3.4 Physical Links	37
3.4.1 Links supported by the basic and adapted piconet physical channel	38
3.4.2 Links supported by the scanning physical channels	39



3.5	Logical Links and Logical Transports.....	39
3.5.1	Casting	41
3.5.2	Scheduling and acknowledgement scheme.....	41
3.5.3	Class of data	42
3.5.4	Asynchronous connection-oriented (ACL)	42
3.5.5	Synchronous connection-oriented (SCO)	43
3.5.6	Extended synchronous connection-oriented (eSCO)....	44
3.5.7	Active slave broadcast (ASB).....	44
3.5.8	Parked slave broadcast (PSB).....	45
3.5.9	Logical links	46
3.5.10	ACL Control Logical Link (ACL-C)	47
3.5.11	User Asynchronous/Isochronous Logical Link (ACL-U) ..	47
3.5.12	User Synchronous/Extended Synchronous Logical Links (SCO-S/eSCO-S)	47
3.6	L2CAP Channels	48
4	Communication Topology	49
4.1	Piconet Topology	49
4.2	Operational Procedures and Modes	51
4.2.1	Inquiry (Discovering) Procedure.....	51
4.2.2	Paging (Connecting) Procedure.....	52
4.2.3	Connected mode.....	52
4.2.4	Hold mode.....	53
4.2.5	Sniff mode	53
4.2.6	Parked state	54
4.2.7	Role switch procedure.....	54



Part B
ACRONYMS & ABBREVIATIONS

1 **List of Acronyms and Abbreviations.....57**

2 **Abbreviations of the Specification Names 65**

Part C
CHANGES FROM BLUETOOTH SPECIFICATION V 1.1

Contents 69

1 **New Features 71**

2 **Changes in Wording 73**

 2.1 IEEE Language Update 73

 2.1.1 Shall 74

 2.1.2 Must.....74

 2.1.3 Will.....74

 2.1.4 Should 74

 2.1.5 May.....75

 2.1.6 Can.....75

 2.2 Nomenclature Changes 75

3 **Structure Changes 77**

4 **Deprecated Specifications 79**

 4.1 Deprecated Features 79

 4.2 Deprecated Profiles.....79


Specification Volume 2
Core System Package
[Controller volume]

Table of Contents	5
--------------------------------	----------

Part A
RADIO SPECIFICATION

Contents	27
1 Scope	29
2 Frequency Bands and Channel Arrangement.....	31
3 Transmitter Characteristics	33
3.1 Modulation Characteristics	34
3.2 Spurious Emissions	35
3.2.1 In-band spurious emission	35
3.3 Radio Frequency Tolerance	36
4 Receiver Characteristics	37
4.1 Actual Sensitivity Level	37
4.2 Interference Performance	37
4.3 Out-of-Band Blocking.....	38
4.4 Intermodulation Characteristics	38
4.5 Maximum Usable Level	39
4.6 Receiver Signal Strength Indicator	39
4.7 Reference Signal Definition	39
5 Appendix A	41
6 Appendix B	43



Part B

BASEBAND SPECIFICATION

Contents	47
1 General Description	53
1.1 Bluetooth Clock	54
1.2 Bluetooth Device Addressing	55
1.2.1 Reserved addresses	55
1.3 Access Codes	56
2 Physical Channels.....	57
2.1 Physical Channel Definition	58
2.2 Basic Piconet Physical Channel.....	58
2.2.1 Master-slave definition	58
2.2.2 Hopping characteristics	59
2.2.3 Time slots	59
2.2.4 Piconet clocks	60
2.2.5 Transmit/receive timing	60
2.3 Adapted Piconet Physical Channel	63
2.3.1 Hopping characteristics	63
2.4 Page Scan Physical Channel.....	64
2.4.1 Clock estimate for paging	64
2.4.2 Hopping characteristics	64
2.4.3 Paging procedure timing	65
2.4.4 Page response timing.....	66
2.5 Inquiry Scan Physical Channel	68
2.5.1 Clock for inquiry.....	68
2.5.2 Hopping characteristics	68
2.5.3 Inquiry procedure timing.....	68
2.5.4 Inquiry response timing	68
2.6 Hop Selection.....	70
2.6.1 General selection scheme.....	70
2.6.2 Selection kernel	74
2.6.3 Adapted hop selection kernel	77
2.6.4 Control word	78
3 Physical Links	83
3.1 Link Supervision.....	83



4	Logical Transports	85
4.1	General	85
4.2	Logical Transport Address (LT_ADDR)	85
4.3	Synchronous Logical Transports	86
4.4	Asynchronous Logical Transport	86
4.5	Transmit/Receive Routines	87
4.5.1	TX Routine	87
4.5.2	RX routine	90
4.5.3	Flow control	91
4.6	Active Slave Broadcast Transport	92
4.7	Parked Slave Broadcast Transport	93
4.7.1	Parked member address (PM_ADDR)	93
4.7.2	Access request address (AR_ADDR)	93
5	Logical Links	95
5.1	Link Control Logical Link (LC)	95
5.2	ACL Control Logical Link (ACL-C)	95
5.3	User Asynchronous/Isochronous Logical Link (ACL-U)	95
5.3.1	Pausing the ACL-U logical link	96
5.4	User Synchronous Data Logical Link (SCO-S)	96
5.5	User Extended Synchronous Data Logical Link (eSCO-S)	96
5.6	Logical Link Priorities	96
6	Packets	97
6.1	General Format	97
6.2	Bit Ordering	97
6.3	Access Code	98
6.3.1	Access code types	98
6.3.2	Preamble	99
6.3.3	Sync word	99
6.3.4	Trailer	102
6.4	Packet Header	103
6.4.1	LT_ADDR	103
6.4.2	TYPE	103
6.4.3	FLOW	104
6.4.4	ARQN	104
6.4.5	SEQN	104
6.4.6	HEC	104



6.5	Packet Types	105
6.5.1	Common packet types.....	106
6.5.2	SCO packets	109
6.5.3	eSCO packets	110
6.5.4	ACL packets	111
6.6	Payload Format.....	112
6.6.1	Synchronous data field.....	112
6.6.2	Asynchronous data field	113
6.7	Packet Summary.....	116
7	Bitstream Processing	117
7.1	Error Checking	118
7.1.1	HEC generation.....	118
7.1.2	CRC generation.....	119
7.2	Data Whitening.....	121
7.3	Error Correction.....	122
7.4	FEC Code: Rate 1/3.....	122
7.5	FEC Code: Rate 2/3.....	123
7.6	ARQ Scheme	124
7.6.1	Unnumbered ARQ.....	124
7.6.2	Retransmit filtering	127
7.6.3	Flushing payloads	130
7.6.4	Multi-slave considerations	130
7.6.5	Broadcast packets.....	130
8	Link Controller Operation.....	133
8.1	Overview of States	133
8.2	Standby State	134
8.3	Connection Establishment Substates	134
8.3.1	Page scan substate.....	134
8.3.2	Page substate	136
8.3.3	Page response substates	139
8.4	Device Discovery Substates	143
8.4.1	Inquiry scan substate	144
8.4.2	Inquiry substate	145
8.4.3	Inquiry response substate	147
8.5	Connection State.....	148



8.6	Active Mode	149
8.6.1	Polling in the active mode	150
8.6.2	SCO	150
8.6.3	eSCO	151
8.6.4	Broadcast scheme	154
8.6.5	Role switch.....	155
8.6.6	Scatternet.....	157
8.6.7	Hop sequence switching	158
8.6.8	Channel classification and channel map selection	161
8.6.9	Power Management.....	162
8.7	Sniff Mode.....	163
8.7.1	Sniff Transition Mode	164
8.8	Hold Mode	165
8.9	Park State	165
8.9.1	Beacon train	166
8.9.2	Beacon access window.....	168
8.9.3	Parked slave synchronization	169
8.9.4	Parking	170
8.9.5	Master-initiated unparking.....	171
8.9.6	Slave-initiated unparking.....	171
8.9.7	Broadcast scan window	172
8.9.8	Polling in the park state.....	172
9	Audio.....	173
9.1	LOG PCM CODEC	173
9.2	CVSD CODEC	173
9.3	Error Handling.....	176
9.4	General Audio Requirements.....	176
9.4.1	Signal levels	176
9.4.2	CVSD audio quality	176
10	List of Figures	177
11	List of Tables	181
Appendix A:		
General Audio Recommendations		182
Appendix B:		
Timers		185
Appendix C:		
Recommendations for AFH Operation in Park, Hold and Sniff		187

**Part C****LINK MANAGER PROTOCOL**

Contents	191
1 Introduction	195
2 General Rules	197
2.1 Message Transport	197
2.2 Synchronization	197
2.3 Packet Format.....	198
2.4 Transactions.....	199
2.4.1 LMP Response Timeout.....	200
2.5 Error Handling	200
2.5.1 Transaction collision resolution	201
2.6 Procedure Rules	201
2.7 General Response Messages.....	202
2.8 LMP Message Constraints	202
3 Device Features.....	203
3.1 General Description	203
3.2 Feature Definitions	203
3.3 Feature Mask Definition	207
3.4 Link Manager Interoperability policy.....	208
4 Procedure Rules.....	209
4.1 Connection Control	209
4.1.1 Connection establishment	209
4.1.2 Detach	210
4.1.3 Power control	211
4.1.4 Adaptive frequency hopping.....	213
4.1.5 Channel classification.....	216
4.1.6 Link supervision.....	218
4.1.7 Channel quality driven data rate change (CQDDR)	219
4.1.8 Quality of service (QoS)	220
4.1.9 Paging scheme parameters	222
4.1.10 Control of multi-slot packets	223
4.2 Security	224
4.2.1 Authentication.....	224
4.2.2 Pairing	226
4.2.3 Change link key	229
4.2.4 Change current link key type.....	230
4.2.5 Encryption	232
4.2.6 Request supported encryption key size	236



4.3	Informational Requests	237
4.3.1	Timing accuracy	237
4.3.2	Clock offset	238
4.3.3	LMP version	238
4.3.4	Supported features	239
4.3.5	Name request	241
4.4	Role Switch	242
4.4.1	Slot offset	242
4.4.2	Role switch	243
4.5	Modes of Operation	245
4.5.1	Hold mode	245
4.5.2	Park state	247
4.5.3	Sniff mode	253
4.6	Logical Transports	256
4.6.1	SCO logical transport	256
4.6.2	eSCO logical transport	259
4.7	Test Mode	264
4.7.1	Activation and deactivation of test mode	264
4.7.2	Control of test mode	265
4.7.3	Summary of test mode PDUs	266
5	Summary	269
5.1	PDU Summary	269
5.2	Parameter Definitions	277
5.3	Default Values	284
6	List of Figures	285
7	List of Tables	289



Part D

ERROR CODES

Contents	293
1 Overview of Error Codes	295
1.1 Usage Descriptions	295
1.2 HCI Command Errors	295
1.3 List of Error Codes	296
2 Error Code Descriptions	299
2.1 Unknown HCI Command (0X01)	299
2.2 Unknown Connection Identifier (0X02)	299
2.3 Hardware Failure (0X03)	299
2.4 Page Timeout (0X04)	299
2.5 Authentication Failure (0X05)	299
2.6 PIN Missing (0X06)	299
2.7 Memory Capacity Exceeded (0X07)	299
2.8 Connection Timeout (0X08)	300
2.9 Connection Limit Exceeded (0X09)	300
2.10 Synchronous Connection Limit to a Device Exceeded (0X0A)	300
2.11 ACL Connection Already Exists (0X0B)	300
2.12 Command Disallowed (0X0C)	300
2.13 Connection Rejected due to Limited Resources (0X0D)	300
2.14 Connection Rejected due to Security Reasons (0X0E)	300
2.15 Connection Rejected due to Unacceptable BD_ADDR (0X0F)	301
2.16 Connection Accept Timeout Exceeded (0X10)	301
2.17 Unsupported Feature or Parameter Value (0X11)	301
2.18 Invalid HCI Command Parameters (0X12)	301
2.19 Remote User Terminated Connection (0X13)	301
2.20 Remote Device Terminated Connection due to Low Resources (0X14)	302
2.21 Remote Device Terminated Connection due to Power Off (0X15)	302
2.22 Connection Terminated by Local Host (0X16)	302
2.23 Repeated Attempts (0X17)	302
2.24 Pairing not Allowed (0X18)	302
2.25 Unknown LMP PDU (0X19)	302
2.26 Unsupported Remote Feature (0X1A)	302
2.27 SCO Offset Rejected (0X1B)	302
2.28 SCO Interval Rejected (0X1C)	303
2.29 SCO Air Mode Rejected (0X1D)	303
2.30 Invalid LMP Parameters (0X1E)	303
2.31 Unspecified Error (0X1F)	303
2.32 Unsupported LMP Parameter Value (0X20)	303



2.33	Role Change Not Allowed (0X21)	303
2.34	LMP Response Timeout (0X22)	303
2.35	LMP Error Transaction Collision (0X23)	304
2.36	LMP PDU Not Allowed (0X24)	304
2.37	Encryption Mode Not Acceptable (0X25)	304
2.38	Link Key Can Not be Changed (0X26)	304
2.39	Requested Qos Not Supported (0X27)	304
2.40	Instant Passed (0X28)	304
2.41	Pairing with Unit Key Not Supported (0X29)	304
2.42	Different Transaction Collision (0x2a)	304
2.43	QoS Unacceptable Parameter (0X2C)	304
2.44	QoS Rejected (0X2D)	305
2.45	Channel Classification Not Supported (0X2E)	305
2.46	Insufficient Security (0X2F)	305
2.47	Parameter out of Mandatory Range (0X30)	305
2.48	Role Switch Pending (0X32)	305
2.49	Reserved Slot Violation (0X34)	305
2.50	Role Switch Failed (0X35)	305



Part E

HOST CONTROLLER INTERFACE FUNCTIONAL SPECIFICATION

Contents	309
1 Introduction	317
1.1 Lower Layers of the Bluetooth Software Stack	317
2 Overview of Host Controller Transport Layer	319
3 Overview of Commands and Events	321
3.1 Generic Events	322
3.2 Device Setup	322
3.3 Controller Flow Control	323
3.4 Controller Information	323
3.5 Controller Configuration	324
3.6 Device Discovery	325
3.7 Connection Setup	327
3.8 Remote Information	329
3.9 Synchronous Connections	330
3.10 Connection State	331
3.11 Piconet Structure	332
3.12 Quality of Service	333
3.13 Physical Links	334
3.14 Host Flow Control	335
3.15 Link Information	336
3.16 Authentication and Encryption	337
3.17 Testing	339
3.18 Alphabetical List of Commands and Events	340
4 HCI Flow Control	345
4.1 Host to Controller Data Flow Control	345
4.2 Controller to Host Data Flow Control	346
4.3 Disconnection Behaviour	347
4.4 Command Flow Control	347
4.5 Command Error Handling	348
5 HCI Data Formats	349
5.1 Introduction	349
5.2 Data and Parameter Formats	349
5.3 Connection Handles	350
5.4 Exchange of HCI-Specific Information	351
5.4.1 HCI Command Packet	351
5.4.2 HCI ACL Data Packets	353
5.4.3 HCI Synchronous Data Packets	355
5.4.4 HCI Event Packet	356



6	HCI Configuration Parameters	357
6.1	Scan Enable	357
6.2	Inquiry Scan Interval	357
6.3	Inquiry Scan Window	358
6.4	Inquiry Scan Type	358
6.5	Inquiry Mode	358
6.6	Page Timeout.....	359
6.7	Connection Accept Timeout.....	359
6.8	Page Scan Interval	360
6.9	Page Scan Window	360
6.10	Page Scan Period Mode.....	360
6.11	Page Scan Type	361
6.12	Voice Setting.....	361
6.13	PIN Type	362
6.14	Link Key	362
6.15	Authentication Enable.....	362
6.16	Encryption Mode.....	363
6.17	Failed Contact Counter	364
6.18	Hold Mode Activity	364
6.19	Link Policy Settings.....	365
6.20	Flush Timeout	366
6.21	Num Broadcast Retransmissions	366
6.22	Link Supervision Timeout.....	367
6.23	Synchronous Flow Control Enable	367
6.24	Local Name.....	368
6.25	Class Of Device	368
6.26	Supported Commands	369
7	HCI Commands and Events	373
7.1	Link Control Commands	373
7.1.1	Inquiry Command.....	373
7.1.2	Inquiry Cancel Command.....	375
7.1.3	Periodic Inquiry Mode Command.....	376
7.1.4	Exit Periodic Inquiry Mode Command.....	379
7.1.5	Create Connection Command.....	380
7.1.6	Disconnect Command.....	383
7.1.7	Create Connection Cancel Command	384
7.1.8	Accept Connection Request Command	386
7.1.9	Reject Connection Request Command	388
7.1.10	Link Key Request Reply Command	389
7.1.11	Link Key Request Negative Reply Command	391
7.1.12	PIN Code Request Reply Command	392



7.1.13	PIN Code Request Negative Reply Command	394
7.1.14	Change Connection Packet Type Command	395
7.1.15	Authentication Requested Command.....	398
7.1.16	Set Connection Encryption Command	399
7.1.17	Change Connection Link Key Command	400
7.1.18	Master Link Key Command	401
7.1.19	Remote Name Request Command	402
7.1.20	Remote Name Request Cancel Command	404
7.1.21	Read Remote Supported Features Command.....	405
7.1.22	Read Remote Extended Features Command	406
7.1.23	Read Remote Version Information Command.....	407
7.1.24	Read Clock Offset Command.....	408
7.1.25	Read LMP Handle Command	409
7.1.26	Setup Synchronous Connection Command	411
7.1.27	Accept Synchronous Connection Request Command	415
7.1.28	Reject Synchronous Connection Request Command	419
7.2	Link Policy Commands.....	420
7.2.1	Hold Mode Command	420
7.2.2	Sniff Mode Command.....	422
7.2.3	Exit Sniff Mode Command.....	425
7.2.4	Park State Command	426
7.2.5	Exit Park State Command	428
7.2.6	QoS Setup Command	429
7.2.7	Role Discovery Command.....	431
7.2.8	Switch Role Command.....	432
7.2.9	Read Link Policy Settings Command	433
7.2.10	Write Link Policy Settings Command	434
7.2.11	Read Default Link Policy Settings Command	436
7.2.12	Write Default Link Policy Settings Command	437
7.2.13	Flow Specification Command	438
7.3	Controller & Baseband Commands.....	440
7.3.1	Set Event Mask Command.....	440
7.3.2	Reset Command	442
7.3.3	Set Event Filter Command	443
7.3.4	Flush Command	448
7.3.5	Read PIN Type Command	450
7.3.6	Write PIN Type Command.....	451
7.3.7	Create New Unit Key Command	452



7.3.8	Read Stored Link Key Command.....	453
7.3.9	Write Stored Link Key Command.....	454
7.3.10	Delete Stored Link Key Command.....	456
7.3.11	Write Local Name Command.....	457
7.3.12	Read Local Name Command.....	458
7.3.13	Read Connection Accept Timeout Command.....	459
7.3.14	Write Connection Accept Timeout Command.....	460
7.3.15	Read Page Timeout Command.....	461
7.3.16	Write Page Timeout Command.....	462
7.3.17	Read Scan Enable Command.....	463
7.3.18	Write Scan Enable Command.....	464
7.3.19	Read Page Scan Activity Command.....	465
7.3.20	Write Page Scan Activity Command.....	467
7.3.21	Read Inquiry Scan Activity Command.....	468
7.3.22	Write Inquiry Scan Activity Command.....	469
7.3.23	Read Authentication Enable Command.....	470
7.3.24	Write Authentication Enable Command.....	471
7.3.25	Read Encryption Mode Command.....	472
7.3.26	Write Encryption Mode Command.....	473
7.3.27	Read Class of Device Command.....	474
7.3.28	Write Class of Device Command.....	475
7.3.29	Read Voice Setting Command.....	476
7.3.30	Write Voice Setting Command.....	477
7.3.31	Read Automatic Flush Timeout Command.....	478
7.3.32	Write Automatic Flush Timeout Command.....	479
7.3.33	Read Num Broadcast Retransmissions Command.....	480
7.3.34	Write Num Broadcast Retransmissions Command.....	481
7.3.35	Read Hold Mode Activity Command.....	482
7.3.36	Write Hold Mode Activity Command.....	483
7.3.37	Read Transmit Power Level Command.....	484
7.3.38	Read Synchronous Flow Control Enable Command...	486
7.3.39	Write Synchronous Flow Control Enable Command...	487
7.3.40	Set Controller To Host Flow Control Command.....	488
7.3.41	Host Buffer Size Command.....	489
7.3.42	Host Number Of Completed Packets Command.....	491
7.3.43	Read Link Supervision Timeout Command.....	493
7.3.44	Write Link Supervision Timeout Command.....	494
7.3.45	Read Number Of Supported IAC Command.....	496



7.3.46	Read Current IAC LAP Command	497
7.3.47	Write Current IAC LAP Command.....	498
7.3.48	Read Page Scan Period Mode Command	500
7.3.49	Write Page Scan Period Mode Command	501
7.3.50	Set AFH Host Channel Classification Command	502
7.3.51	Read Inquiry Scan Type Command	503
7.3.52	Write Inquiry Scan Type Command	504
7.3.53	Read Inquiry Mode Command	505
7.3.54	Write Inquiry Mode Command	506
7.3.55	Read Page Scan Type Command	507
7.3.56	Write Page Scan Type Command	508
7.3.57	Read AFH Channel Assessment Mode Command	509
7.3.58	Write AFH Channel Assessment Mode Command	510
7.4	Informational Parameters.....	512
7.4.1	Read Local Version Information Command.....	512
7.4.2	Read Local Supported Commands Command.....	514
7.4.3	Read Local Supported Features Command.....	515
7.4.4	Read Local Extended Features Command	516
7.4.5	Read Buffer Size Command.....	518
7.4.6	Read BD_ADDR Command	520
7.5	Status Parameters.....	521
7.5.1	Read Failed Contact Counter Command	521
7.5.2	Reset Failed Contact Counter Command	523
7.5.3	Read Link Quality Command	524
7.5.4	Read RSSI Command.....	525
7.5.5	Read AFH Channel Map Command	527
7.5.6	Read Clock Command	529
7.6	Testing Commands	531
7.6.1	Read Loopback Mode Command	531
7.6.2	Write Loopback Mode Command.....	532
7.6.3	Enable Device Under Test Mode Command	535
7.7	Events	537
7.7.1	Inquiry Complete Event.....	537
7.7.2	Inquiry Result Event	538
7.7.3	Connection Complete Event.....	540
7.7.4	Connection Request Event.....	541
7.7.5	Disconnection Complete Event	543
7.7.6	Authentication Complete Event.....	544



7.7.7	Remote Name Request Complete Event	545
7.7.8	Encryption Change Event	546
7.7.9	Change Connection Link Key Complete Event	547
7.7.10	Master Link Key Complete Event	548
7.7.11	Read Remote Supported Features Complete Event ...	549
7.7.12	Read Remote Version Information Complete Event ...	550
7.7.13	QoS Setup Complete Event	551
7.7.14	Command Complete Event	553
7.7.15	Command Status Event	554
7.7.16	Hardware Error Event	555
7.7.17	Flush Occurred Event	555
7.7.18	Role Change Event	556
7.7.19	Number Of Completed Packets Event	557
7.7.20	Mode Change Event	558
7.7.21	Return Link Keys Event	560
7.7.22	PIN Code Request Event	561
7.7.23	Link Key Request Event	562
7.7.24	Link Key Notification Event	563
7.7.25	Loopback Command Event	564
7.7.26	Data Buffer Overflow Event	564
7.7.27	Max Slots Change Event	565
7.7.28	Read Clock Offset Complete Event	566
7.7.29	Connection Packet Type Changed Event	567
7.7.30	QoS Violation Event	569
7.7.31	Page Scan Repetition Mode Change Event	570
7.7.32	Flow Specification Complete Event	571
7.7.33	Inquiry Result with RSSI Event	573
7.7.34	Read Remote Extended Features Complete Event ...	575
7.7.35	Synchronous Connection Complete Event	576
7.7.36	Synchronous Connection Changed event	578
8	List of Figures	581
9	List of Tables	583
Appendix A:		
	Deprecated Commands, Events and Configuration Parameters	585

**Part F****MESSAGE SEQUENCE CHARTS**

Contents	595
1 Introduction	599
1.1 Notation	599
1.2 Flow of Control	600
1.3 Example MSC	600
2 Services Without Connection Request	601
2.1 Remote Name Request	601
2.2 One-time Inquiry	602
2.3 Periodic Inquiry	604
3 ACL Connection Establishment and Detachment	607
3.1 Connection Setup	608
4 Optional Activities After ACL Connection Establishment	615
4.1 Authentication Requested	615
4.2 Set Connection Encryption	616
4.3 Change Connection Link Key	617
4.4 Master Link Key	618
4.5 Read Remote Supported Features	620
4.6 Read Remote Extended Features	620
4.7 Read Clock Offset	621
4.8 Read Remote Version Information	621
4.9 QOS Setup	622
4.10 Switch Role	622
5 Synchronous Connection Establishment and Detachment	625
5.1 Synchronous Connection Setup	625
6 Sniff, Hold and Park	631
6.1 Sniff Mode	631
6.2 Hold Mode	632
6.3 Park State	634
7 Buffer Management, Flow Control	637
8 Loopback Mode	639
8.1 Local Loopback Mode	639
8.2 Remote Loopback Mode	641
9 List of Figures	643

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

Contents	647
1 Encryption Sample Data	649
1.1 Generating Kc' from Kc, ...	649
1.2 First Set of Sample Data	652
1.3 Second Set of Sample Data	660
1.4 Third Set of Samples	668
1.5 Fourth Set of Samples	676
2 Frequency Hopping Sample Data	685
2.1 First set	686
2.2 Second set	692
2.3 Third set	698
3 Access Code Sample Data	705
4 HEC and Packet Header Sample Data	709
5 CRC Sample Data	711
6 Complete Sample Packets	713
6.1 Example of DH1 Packet	713
6.2 Example of DM1 Packet	714
7 Whitening Sequence Sample Data	715
8 FEC Sample Data	719
9 Encryption Key Sample Data	721
9.1 Four Tests of E1	721
9.2 Four Tests of E21	726
9.3 Three Tests of E22	728
9.4 Tests of E22 With Pin Augmenting	730
9.5 Four Tests of E3	740



Part H

SECURITY SPECIFICATION

Contents	747
1 Security Overview	749
2 Random Number Generation	751
3 Key Management	753
3.1 Key Types	753
3.2 Key Generation and Initialization	755
3.2.1 Generation of initialization key,	756
3.2.2 Authentication	756
3.2.3 Generation of a unit key	756
3.2.4 Generation of a combination key	757
3.2.5 Generating the encryption key	758
3.2.6 Point-to-multipoint configuration	759
3.2.7 Modifying the link keys	760
3.2.8 Generating a master key	760
4 Encryption	763
4.1 Encryption Key Size Negotiation	764
4.2 Encryption of Broadcast Messages	764
4.3 Encryption Concept	765
4.4 Encryption Algorithm	766
4.4.1 The operation of the cipher	768
4.5 LFSR Initialization	769
4.6 Key Stream Sequence	772
5 Authentication	773
5.1 Repeated Attempts	775
6 The Authentication And Key-Generating Functions	777
6.1 The Authentication Function E1	777
6.2 The Functions Ar and A'r	779
6.2.1 The round computations	779
6.2.2 The substitution boxes "e" and "l"	779
6.2.3 Key scheduling	780
6.3 E2-Key Generation Function for Authentication	781
6.4 E3-Key Generation Function for Encryption	783
7 List of Figures	785
8 List of Tables	787


Specification Volume 3
Core System Package
[Host volume]
Table of Contents 5
Part A
LOGICAL LINK CONTROL AND ADAPTATION PROTOCOL SPECIFICATION
Contents 17

1	Introduction	21
1.1	L2CAP Features	21
1.2	Assumptions	25
1.3	Scope.....	25
1.4	Terminology	26
2	General Operation.....	29
2.1	Channel Identifiers	29
2.2	Operation Between Devices	29
2.3	Operation Between Layers	31
2.4	Modes of Operation	31
3	Data Packet Format.....	33
3.1	Connection-oriented Channel in Basic L2CAP Mode	33
3.2	Connectionless Data Channel in Basic L2CAP Mode	34
3.3	Connection-oriented Channel in Retransmission/Flow Control Modes	35
3.3.1	L2CAP header fields	35
3.3.2	Control field (2 octets)	36
3.3.3	L2CAP SDU length field (2 octets)	38
3.3.4	Information payload field (0 to 65531 octets)	38
3.3.5	Frame check sequence (2 octets).....	39
3.3.6	Invalid frame detection	40
4	Signalling Packet Formats	41
4.1	Command Reject (code 0x01)	43
4.2	Connection Request (code 0x02)	44
4.3	Connection Response (code 0x03)	46
4.4	Configuration Request (code 0x04)	47
4.5	Configuration Response (code 0x05)	50
4.6	Disconnection Request (code 0x06).....	52
4.7	Disconnection Response (code 0x07)	53
4.8	Echo Request (code 0x08)	53
4.9	Echo Response (code 0x09)	54



4.10	Information Request (code 0x0A)	54
4.11	Information Response (code 0x0B).....	55
4.12	Extended Feature Mask	56
5	Configuration Parameter Options.....	57
5.1	Maximum Transmission Unit (MTU).....	57
5.2	Flush Timeout Option	59
5.3	Quality of Service (QoS) Option.....	60
5.4	Retransmission and Flow Control Option.....	64
6	State Machine	67
6.1	General rules for the state machine:	67
6.1.1	CLOSED state	68
6.1.2	WAIT_CONNECT_RSP state	69
6.1.3	WAIT_CONNECT state	69
6.1.4	CONFIG state.....	70
6.1.5	OPEN state	73
6.1.6	WAIT_DISCONNECT state	73
6.2	Timers events.....	74
6.2.1	RTX	74
6.2.2	ERTX.....	75
7	General Procedures	79
7.1	Configuration Process.....	79
7.1.1	Request path	80
7.1.2	Response path	80
7.2	Fragmentation and Recombination	81
7.2.1	Fragmentation of L2CAP PDUs	81
7.2.2	Recombination of L2CAP PDUs.....	82
7.3	Encapsulation of SDUs	83
7.3.1	Segmentation of L2CAP SDUs	83
7.3.2	Reassembly of L2CAP SDUs.....	84
7.3.3	Segmentation and fragmentation	84
7.4	Delivery of Erroneous L2CAP SDUs.....	85
7.5	Operation with Flushing	85
7.6	Connectionless Data Channel.....	86



8	Procedures for Flow Control and Retransmission	87
8.1	Information Retrieval.....	87
8.2	Function of PDU Types for Flow Control and Retransmission...	87
8.2.1	Information frame (I-frame)	87
8.2.2	Supervisory Frame (S-frame).....	87
8.3	Variables and Sequence Numbers	89
8.3.1	Sending peer.....	89
8.3.2	Receiving peer	91
8.4	Retransmission Mode	93
8.4.1	Transmitting frames	93
8.4.2	Receiving I-frames	95
8.4.3	I-frames pulled by the SDU reassembly function	95
8.4.4	Sending and receiving acknowledgements.....	96
8.4.5	Receiving REJ frames.....	97
8.4.6	Waiting acknowledgements	97
8.4.7	Exception conditions	97
8.5	Flow Control Mode.....	99
8.5.1	Transmitting I-frames	99
8.5.2	Receiving I-frames	100
8.5.3	I-frames pulled by the SDU reassembly function	100
8.5.4	Sending and receiving acknowledgements.....	100
8.5.5	Waiting acknowledgements	101
8.5.6	Exception conditions	101
9	List of Figures	103
10	List of Tables	105
Appendix A:		
	Configuration MSCs	107

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

Contents	113
1 Introduction	115
1.1 General Description	115
1.2 Motivation	115
1.3 Requirements	115
1.4 Non-requirements and Deferred Requirements	116
1.5 Conventions	116
1.5.1 Bit And Byte Ordering Conventions	116
2 Overview	117
2.1 SDP Client-Server Interaction	117
2.2 Service Record	118
2.3 Service Attribute	120
2.4 Attribute ID	120
2.5 Attribute Value	121
2.6 Service Class	121
2.6.1 A Printer Service Class Example	122
2.7 Searching for Services	123
2.7.1 UUID	123
2.7.2 Service Search Patterns	124
2.8 Browsing for Services	124
2.8.1 Example Service Browsing Hierarchy	125
3 Data Representation	127
3.1 Data Element	127
3.2 Data Element Type Descriptor	127
3.3 Data Element Size Descriptor	128
3.4 Data Element Examples	129
4 Protocol Description	131
4.1 Transfer Byte Order	131
4.2 Protocol Data Unit Format	131
4.3 Partial Responses and Continuation State	133
4.4 Error Handling	133
4.4.1 SDP_ErrorResponse PDU	134
4.5 ServiceSearch Transaction	135
4.5.1 SDP_ServiceSearchRequest PDU	135
4.5.2 SDP_ServiceSearchResponse PDU	136
4.6 ServiceAttribute Transaction	138
4.6.1 SDP_ServiceAttributeRequest PDU	138
4.6.2 SDP_ServiceAttributeResponse PDU	140



4.7	ServiceSearchAttribute Transaction	141
4.7.1	SDP_ServiceSearchAttributeRequest PDU	141
4.7.2	SDP_ServiceSearchAttributeResponse PDU	143
5	Service Attribute Definitions	145
5.1	Universal Attribute Definitions	145
5.1.1	ServiceRecordHandle Attribute	145
5.1.2	ServiceClassIDList Attribute	146
5.1.3	ServiceRecordState Attribute	146
5.1.4	ServiceID Attribute	146
5.1.5	ProtocolDescriptorList Attribute	147
5.1.6	BrowseGroupList Attribute	148
5.1.7	LanguageBaseAttributeIDList Attribute	148
5.1.8	ServiceInfoTimeToLive Attribute	149
5.1.9	ServiceAvailability Attribute	150
5.1.10	BluetoothProfileDescriptorList Attribute	150
5.1.11	DocumentationURL Attribute	151
5.1.12	ClientExecutableURL Attribute	151
5.1.13	IconURL Attribute	152
5.1.14	ServiceName Attribute	152
5.1.15	ServiceDescription Attribute	153
5.1.16	ProviderName Attribute	153
5.1.17	Reserved Universal Attribute IDs	153
5.2	ServiceDiscoveryServer Service Class Attribute Definitions ...	154
5.2.1	ServiceRecordHandle Attribute	154
5.2.2	ServiceClassIDList Attribute	154
5.2.3	VersionNumberList Attribute	154
5.2.4	ServiceDatabaseState Attribute	155
5.2.5	Reserved Attribute IDs	155
5.3	BrowseGroupDescriptor Service Class Attribute Definitions ...	155
5.3.1	ServiceClassIDList Attribute	155
5.3.2	GroupID Attribute	156
5.3.3	Reserved Attribute IDs	156
Appendix A:		
Background Information		157
Appendix B:		
Example SDP Transactions		158



Part C

GENERIC ACCESS PROFILE

Contents	173
Foreword	177
1 Introduction	179
1.1 Scope	179
1.2 Symbols and conventions	179
1.2.1 Requirement status symbols	179
1.2.2 Signaling diagram conventions	180
1.2.3 Notation for timers and counters	180
2 Profile overview	181
2.1 Profile stack	181
2.2 Configurations and roles	181
2.3 User requirements and scenarios	182
2.4 Profile fundamentals	183
2.5 Conformance	183
3 User interface aspects	185
3.1 The user interface level	185
3.2 Representation of Bluetooth parameters	185
3.2.1 Bluetooth device address (BD_ADDR)	185
3.2.2 Bluetooth device name (the user-friendly name)	185
3.2.3 Bluetooth passkey (Bluetooth PIN)	186
3.2.4 Class of Device	187
3.3 Pairing	188
4 Modes	189
4.1 Discoverability modes	189
4.1.1 Non-discoverable mode	190
4.1.2 Limited discoverable mode	190
4.1.3 General discoverable mode	191
4.2 Connectability modes	193
4.2.1 Non-connectable mode	193
4.2.2 Connectable mode	193
4.3 Pairing modes	195
4.3.1 Non-pairable mode	195
4.3.2 Pairable mode	195



5	Security aspects.....	197
5.1	Authentication	197
5.1.1	Purpose	197
5.1.2	Term on UI level	197
5.1.3	Procedure	198
5.1.4	Conditions	198
5.2	Security modes	198
5.2.1	Security mode 1 (non-secure).....	200
5.2.2	Security mode 2 (service level enforced security).....	200
5.2.3	Security modes 3 (link level enforced security).....	200
6	Idle mode procedures.....	201
6.1	General inquiry	201
6.1.1	Purpose	201
6.1.2	Term on UI level	201
6.1.3	Description	202
6.1.4	Conditions	202
6.2	Limited inquiry.....	202
6.2.1	Purpose	202
6.2.2	Term on UI level	203
6.2.3	Description	203
6.2.4	Conditions	203
6.3	Name discovery	204
6.3.1	Purpose	204
6.3.2	Term on UI level	204
6.3.3	Description	204
6.3.4	Conditions	205
6.4	Device discovery.....	205
6.4.1	Purpose	205
6.4.2	Term on UI level	205
6.4.3	Description	206
6.4.4	Conditions	206
6.5	Bonding.....	206
6.5.1	Purpose	206
6.5.2	Term on UI level	206
6.5.3	Description	207
6.5.4	Conditions	208



7	Establishment procedures	209
7.1	Link establishment	209
7.1.1	Purpose	209
7.1.2	Term on UI level	209
7.1.3	Description	210
7.1.4	Conditions	211
7.2	Channel establishment.....	212
7.2.1	Purpose	212
7.2.2	Term on UI level	212
7.2.3	Description	212
7.2.4	Conditions	213
7.3	Connection establishment.....	214
7.3.1	Purpose	214
7.3.2	Term on UI level	214
7.3.3	Description	214
7.3.4	Conditions	215
7.4	Establishment of additional connection.....	215
8	Definitions	217
8.1	General definitions	217
8.2	Connection-related definitions.....	217
8.3	Device-related definitions.....	218
8.4	Procedure-related definitions	219
8.5	Security-related definitions.....	219
9	Appendix A (Normative):	
	Timers and constants	221
10	Appendix B (Informative):	
	Information flows of related procedures.....	223
11	References	227



Part D

TEST SUPPORT

Contents	231
1 Test Methodology.....	233
1.1 Test Scenarios	233
1.1.1 Test setup.....	233
1.1.2 Transmitter Test	234
1.1.3 LoopBack test	239
1.1.4 Pause test	242
1.2 References	242
2 Test Control Interface (TCI).....	243
2.1 Introduction	243
2.1.1 Terms used	243
2.1.2 Usage of the interface	243
2.2 TCI Configurations	244
2.2.1 Bluetooth RF requirements	244
2.2.2 Bluetooth protocol requirements	245
2.2.3 Bluetooth profile requirements	246
2.3 TCI Configuration and Usage	247
2.3.1 Transport layers	247
2.3.2 Baseband and link manager qualification	248
2.3.3 HCI qualification	250



4.2 SECURITY

4.2.1 Authentication

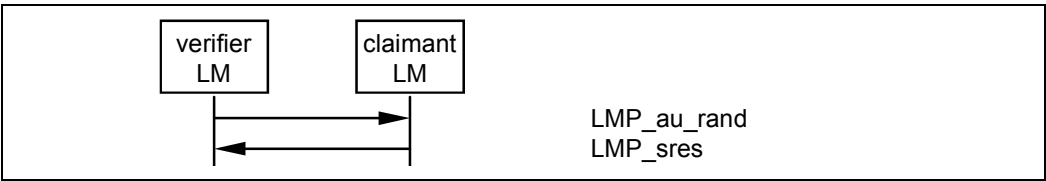
The authentication procedure is based on a challenge-response scheme as described in [Part H] Section 5 on page 773. The verifier sends an LMP_au_rand PDU that contains a random number (the challenge) to the claimant. The claimant calculates a response, that is a function of this challenge, the claimant's BD_ADDR and a secret key. The response is sent back to the verifier, that checks if the response was correct or not. The response shall be calculated as described in [Part H] Section 6.1 on page 777. A successful calculation of the authentication response requires that two devices share a secret key. This key is created as described in Section 4.2.2 on page 226. Both the master and the slave can be verifiers.

M/O	PDU	Contents
M	LMP_au_rand	random number
M	LMP_sres	authentication response

Table 4.11: PDUs used for authentication.

4.2.1.1 Claimant has link key

If the claimant has a link key associated with the verifier, it shall calculate the response and sends it to the verifier with LMP_sres. The verifier checks the response. If the response is not correct, the verifier can end the connection by sending an LMP_detach PDU with the error code *authentication failure*, see Section 4.1.2 on page 210.



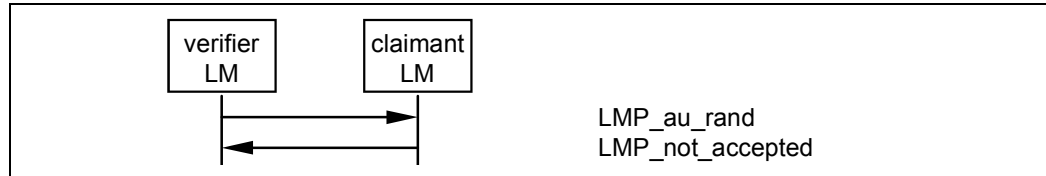
Sequence 20: Authentication. Claimant has link key.

Upon reception of an LMP_au_rand, an LM shall reply with LMP_sres before initiating its own authentication.
Note: there can be concurrent requests caused by the master and slave simultaneously initiating an authentication. The procedures in Section 2.5.1 on page 201 assures that devices will not have different Authenticated Cyphering Offset (ACO, see [Part H] Section 6.1 on page 777) when they calculate the encryption key.



4.2.1.2 Claimant has no link key

If the claimant does not have a link key associated with the verifier it shall send an LMP_not_accepted PDU with the error code *key missing* after receiving an LMP_aud_rand PDU.



Sequence 21: Authentication fails. Claimant has no link key.

4.2.1.3 Repeated attempts

The scheme described in [\[Part H\] Section 5.1 on page 775](#) shall be applied when an authentication fails. This will prevent an intruder from trying a large number of keys in a relatively short time.

4.2.2 Pairing

When two devices do not have a common link key an initialization key (K_{init}) shall be created based on a PIN, and a random number and a BD_ADDR. K_{init} shall be created as specified in [\[Part H\] Section 6.3 on page 781](#). When both devices have calculated K_{init} the link key shall be created, and a mutual authentication is performed. The pairing procedure starts with a device sending an LMP_in_rand PDU; this device is referred to as the "initiating LM" or "initiator" in [Section 4.2.2.1 on page 226](#) - [Section 4.2.2.5 on page 228](#). The other device is referred to as the "responding LM" or "responder". The PDUs used in the pairing procedure are:

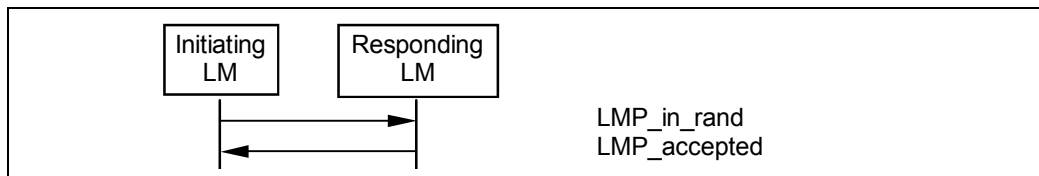
M/O	PDU	Contents
M	LMP_in_rand	random number
M	LMP_au_rand	random number
M	LMP_sres	authentication response
M	LMP_comb_key	random number
M	LMP_unit_key	key

Table 4.12: PDUs used for pairing

All sequences described in [Section 4 on page 209](#), including the mutual authentication after the link key has been created, shall form a single transaction. The transaction ID from the first LMP_in_rand shall be used for all subsequent sequences.

4.2.2.1 Responder accepts pairing

When the initiator sends an LMP_in_rand PDU and the responder shall reply with an LMP_accepted PDU. Both devices shall then calculate K_{init} based on the BD_ADDR of the responder and the procedure continues with creation of the link key; see [Section 4.2.2.4 on page 228](#).

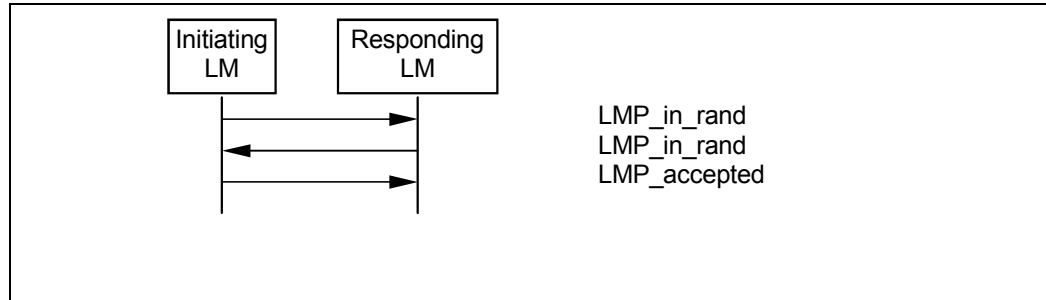


Sequence 22: Pairing accepted. Responder has a variable PIN. Initiator has a variable or fixed PIN.



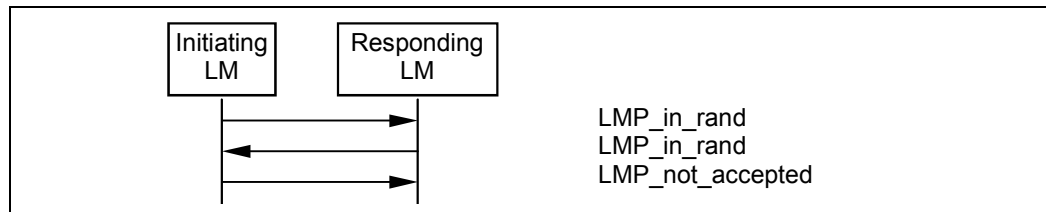
4.2.2.2 Responder has a fixed PIN

If the responder has a fixed PIN it shall generate a new random number and send it back in an LMP_in_rand PDU. If the initiator has a variable PIN it shall accept the LMP_in_rand PDU and shall respond with an LMP_accepted PDU. Both sides shall then calculate K_{init} based on the last IN_RANDOM and the BD_ADDR of the initiator. The procedure continues with creation of the link key; see [Section 4.2.2.4 on page 228](#).



Sequence 23: Responder has a fixed PIN and initiator has a variable PIN.

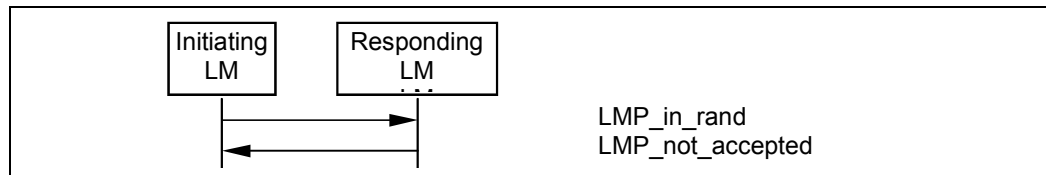
If the responder has a fixed PIN and the initiator also has a fixed PIN, the second LMP_in_rand shall be rejected by the initiator sending an LMP_not_accepted PDU with the error code *pairing not allowed*.



Sequence 24: Both devices have a fixed PIN.

4.2.2.3 Responder rejects pairing

If the responder rejects pairing it shall send an LMP_not_accepted PDU with the error code *pairing not allowed* after receiving an LMP_in_rand PDU.



Sequence 25: Responder rejects pairing.



4.2.2.4 Creation of the link key

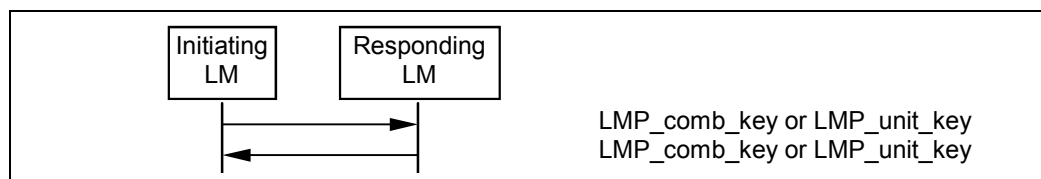
When K_{init} is calculated in both devices the link key shall be created. This link key will be used in the authentication between the two devices for all subsequent connections until it is changed; see [Section 4.2.3 on page 229](#) and [Section 4.2.4 on page 230](#). The link key created in the pairing procedure will either be a combination key or one of the device's unit keys. The following rules shall apply to the selection of the link key:

- if one device sends an LMP_unit_key PDU and the other device sends LMP_comb_key, the unit key will be the link key.
- if both devices send an LMP_unit_key PDU, the master's unit key will be the link key.
- if both devices send an LMP_comb_key PDU, the link key shall be calculated as described in [\[Part H\] Section 3.2 on page 755](#).

The content of the LMP_unit_key PDU is the unit key bitwise XORed with K_{init} . The content of the LMP_comb_key PDU is LK_RAND bitwise XORed with K_{init} . Any device configured to use a combination key shall store the link key.

The use of unit keys is deprecated since it is implicitly insecure.

When the link key, combination or unit key, has been created mutual authentication shall be performed to confirm that the same link key has been created in both devices. The first authentication in the mutual authentication is performed with the initiator as the verifier. When finalized an authentication in the reverse direction is performed.



Sequence 26: Creation of the link key.

4.2.2.5 Repeated attempts

When the authentication after creation of the link key fails because of an incorrect authentication response, the same scheme as in [Section 4.2.1.3 on page 225](#) shall be used. This prevents an intruder from trying a large number of different PINs in a relatively short time.

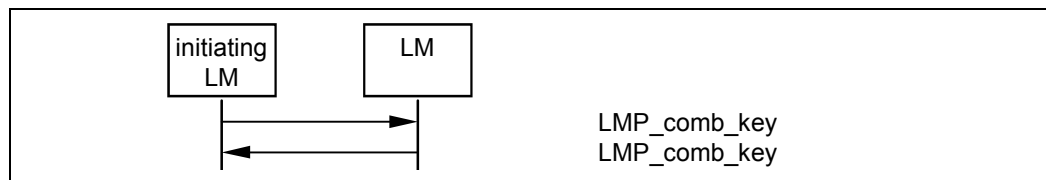
4.2.3 Change link key

If the link key is derived from combination keys and the current link is the semi-permanent link key, the link key can be changed. If the link key is a unit key, the devices shall go through the pairing procedure in order to change the link key. The contents of the LMP_comb_key PDU is protected by a bitwise XOR with the current link key.

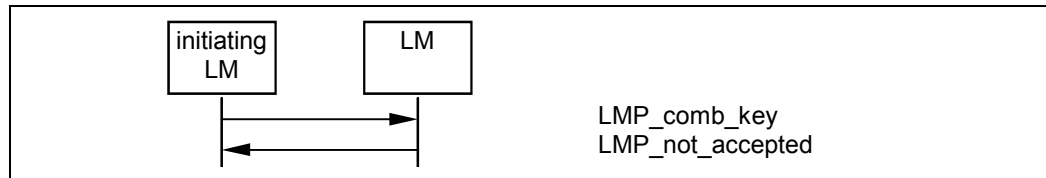
M/O	PDU	Contents
M	LMP_comb_key	random number

Table 4.13: PDUs used for change of link key.

All sequences described in [Section 4.2.3](#), including the mutual authentication after the link key has been changed, shall form a single transaction. The transaction ID from the first LMP_comb_key PDU shall be used for all subsequent sequences.



Sequence 27: Successful change of the link key.



Sequence 28: Change of the link key not possible since the other device uses a unit key.

If the change of link key is successful the new link key shall be stored and the old link key shall be discarded. The new link key shall be used as link key for all the following connections between the two devices until the link key is changed again. The new link key also becomes the current link key. It will remain the current link key until the link key is changed again, or until a temporary link key is created, see [Section 4.2.4 on page 230](#).

When the new link key has been created mutual authentication shall be performed to confirm that the same link key has been created in both devices. The first authentication in the mutual authentication is performed with the device that initiated change link key as verifier. When finalized an authentication in the reverse direction is performed.

4.2.4 Change current link key type

The current link key can be a semi-permanent link key or a temporary link key. It may be changed temporarily, but the change shall only be valid for the current connection, see [\[Part H\] Section 3.1 on page 753](#). Changing to a temporary link key is necessary if the piconet is to support encrypted broadcast. The current link key may not be changed before the connection establishment procedure has completed. This feature is only supported if broadcast encryption is supported as indicated by the LMP features mask.

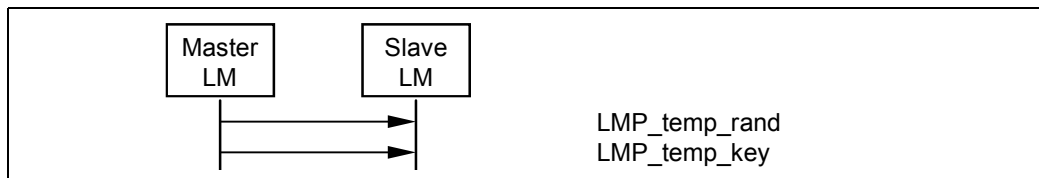
M/O	PDU	Contents
O(23)	LMP_temp_rand	random number
O(23)	LMP_temp_key	key
O(23)	LMP_use_semi_permanent_key	-

Table 4.14: PDUs used to change the current link key.

4.2.4.1 Change to a temporary link key

The master starts by creating the master key K_{master} as specified in [Security Specification \(EQ 4\), on page 760](#). Then the master shall generate a random number, RAND, and shall send it to the slave in an LMP_temp_rand PDU. Both sides then calculate an overlay denoted OVL as $\text{OVL} = \bar{E}_{22}(\text{current link key}, \text{RAND}, 16)$. The master shall then send K_{master} protected by a modulo-2 addition with OVL to the slave in an LMP_temp_key PDU. The slave calculates K_{master} based on OVL, that becomes the current link key. It shall be the current link key until the devices fall back to the semi-permanent link key, see [section 4.2.4.2 on page 231](#).

Note: the terminology in this section is the same as used in [\[Part H\] Section 3.2.8 on page 760](#).



Sequence 29: Change to a temporary link key.

All sequences described in [Section 4.2.4.1 on page 230](#), including the mutual authentication after K_{master} has been created, shall form a single transaction. The transaction ID shall be set to 0.

When the devices have changed to the temporary key, a mutual authentication shall be made to confirm that the same link key has been created in both devices. The first authentication in the mutual authentication shall be per-

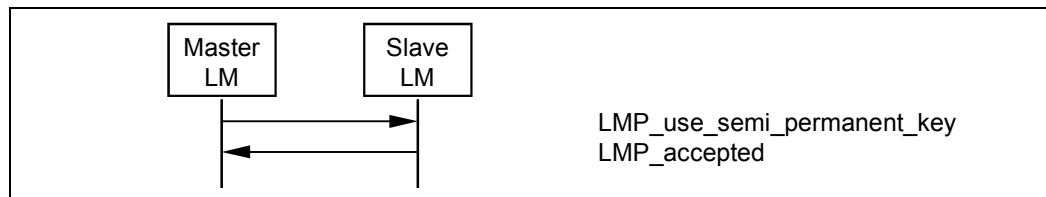


formed with the master as verifier. When finalized an authentication in the reverse direction is performed.

Should the mutual authentication fail at either side, the LM of the verifier should start the detach procedure. This will allow the procedure to succeed even though one of the devices may be erroneous.

4.2.4.2 Make the semi-permanent link key the current link key

After the current link key has been changed to K_{master} , this change can be undone and the semi-permanent link key becomes the current link key again. If encryption is used on the link, the procedure to go back to the semi-permanent link key shall be immediately followed by the master stopping encryption using the procedure described in [Section 4.2.5.4 on page 235](#). Encryption may be restarted by the master according to the procedures in [section 4.2.5.1 on page 232](#) subsection 3. This is to assure that encryption with encryption parameters known by other devices in the piconet is not used when the semi-permanent link key is the current link key.



Sequence 30: Link key changed to the semi-permanent link key.

4.2.5 Encryption

If at least one authentication has been performed encryption may be used. In order for the master to use the same encryption parameters for all slaves in the piconet it shall issue a temporary key, K_{master} . The master shall make this key the current link key for all slaves in the piconet before encryption is started, see [Section 4.2.4 on page 230](#). This is required if broadcast packets are to be encrypted.

M/O	PDU	Contents
O	LMP_encryption_mode_req	encryption mode
O	LMP_encryption_key_size_req	key size
O	LMP_start_encryption_req	random number
O	LMP_stop_encryption_req	-

Table 4.15: PDUs used for handling encryption.

All sequences described in [Section 4.2.5](#) shall form a single transaction. The transaction ID from the LMP_encryption_mode_req PDU shall be used for all subsequent sequences.

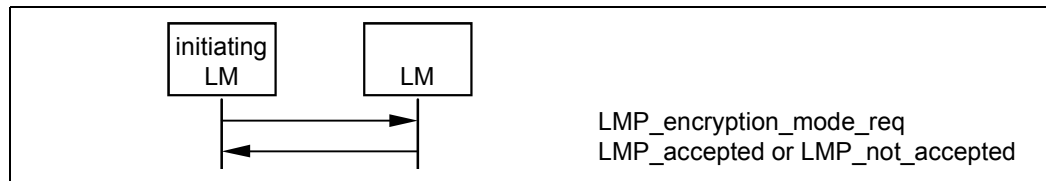
4.2.5.1 Encryption mode

The master and the slave must agree upon whether to use encryption (encryption mode=1 in Lmp_encryption_mode_req) or not (encryption mode=0). If the semi-permanent key is used (Key_Flag=0x00) encryption shall only apply to point-to-point packets. If the master link key is used (Key_Flag=0x01) encryption shall apply to both point-to-point packets and broadcast packets. If master and slave agree on the encryption mode, the master continues to give more detailed information about the encryption.

Devices should never send LMP_encryption_mode_req with an encryption mode value of 2 however for backwards compatibility if the LMP_encryption_mode_req is received with an encryption mode value of 2 then it should be treated the same as an encryption mode value of 1.

The initiating LM shall pause traffic on the ACL-U logical link (see [Baseband Specification, Section 5.3.1, on page 96](#)). The initiating device shall then send the LMP_encryption_mode_req PDU. If the responding device accepts the change in encryption mode then it shall complete the transmission of the current packet on the ACL logical transport and shall then suspend transmission on the ACL-U logical link. The responding device shall then send the LMP_accepted PDU.

ACL-U logical link traffic shall only be resumed after the attempt to encrypt or decrypt the logical transport is completed i.e. at the end of Sequence [31](#), [32](#) or [33](#).

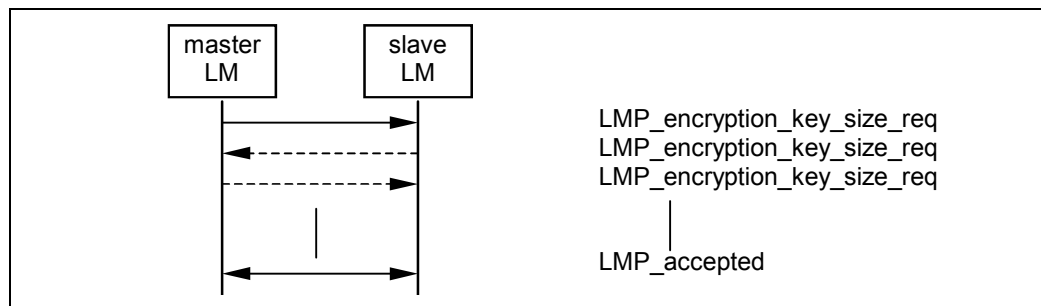


Sequence 31: Negotiation for encryption mode.

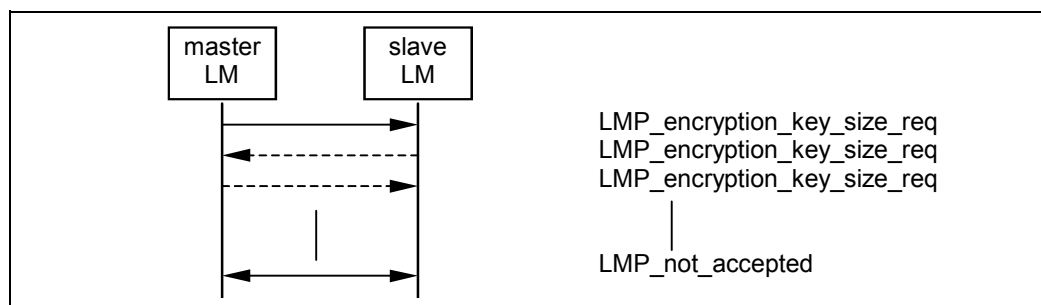
After a device has sent an LMP_encryption_mode_req PDU it shall not send an LMP_aud_rand PDU before encryption is started. After a device has received an LMP_encryption_mode_req PDU and sent an LMP_accepted PDU it shall not send an LMP_aud_rand PDU before encryption is started. If an LMP_aud_rand PDU is sent violating these rules, the claimant shall respond with an LMP_not_accepted PDU with the error code *PDU not allowed*. This assures that devices will not have different ACOs when they calculate the encryption key. If the encryption mode is not accepted or the encryption key size negotiation results in disagreement the devices may send an LMP_aud_rand PDU again.

4.2.5.2 Encryption key size

Note: this section uses the same terms as in [\[Part H\] Section 4.1 on page 764](#). The master sends an LMP_encryption_key_size_req PDU including the suggested key size $L_{sug, m}$, that is initially equal to $L_{max, m}$. If $L_{min, s} \leq L_{sug, m}$ and the slave supports $L_{sug, m}$ it shall respond with an LMP_accepted PDU and $L_{sug, m}$ shall be used as the key size. If both conditions are not fulfilled the slave sends back an LMP_encryption_key_size_req PDU including the slave's suggested key size $L_{sug, s}$. This value shall be the slave's largest supported key size that is less than $L_{sug, m}$. Then the master performs the corresponding test on the slave's suggestion. This procedure is repeated until a key size agreement is reached or it becomes clear that no such agreement can be reached. If an agreement is reached a device sends an LMP_accepted PDU and the key size in the last LMP_encryption_key_size_req PDU shall be used. After this, encryption is started; see [Section 4.2.5.3 on page 234](#). If an agreement is not reached a device sends an LMP_not_accepted PDU with the error code *unsupported parameter value* and the devices shall not communicate using encryption.



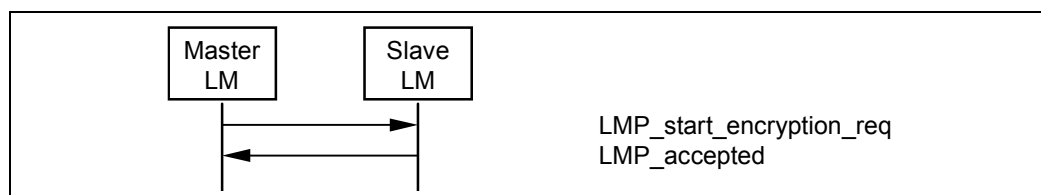
Sequence 32: Encryption key size negotiation successful.



Sequence 33: Encryption key size negotiation failed.

4.2.5.3 Start encryption

To start encryption, the master issues the random number EN_RANDOM and calculates the encryption key. See [\[Part H\] Section 3.2.5 on page 758](#). The random number shall be the same for all slaves in the piconet when broadcast encryption is used. The master then sends an LMP_start_encryption_req PDU, that includes EN_RANDOM. The slave shall calculate the encryption key when this message is received and shall acknowledge with an LMP_accepted PDU.



Sequence 34: Start of encryption.

Starting encryption shall be performed in three steps:

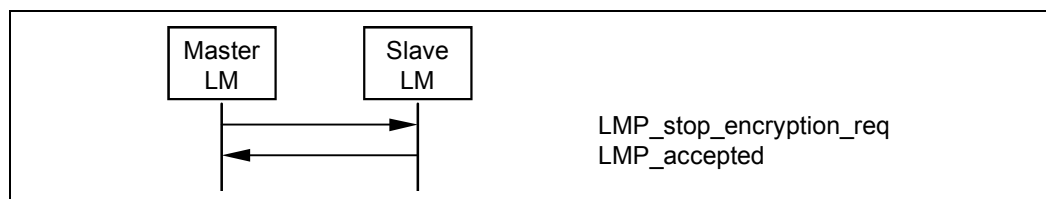
1. Master is configured to transmit unencrypted packets and to receive encrypted packets.
2. Slave is configured to transmit and receive encrypted packets.
3. Master is configured to transmit and receive encrypted packets.



Between step 1 and step 2, master-to-slave transmission is possible. This is when an LMP_start_encryption_req PDU is transmitted. Step 2 is triggered when the slave receives this message. Between step 2 and step 3, slave-to-master transmission is possible. This is when an LMP_accepted PDU is transmitted. Step 3 is triggered when the master receives this message.

4.2.5.4 Stop encryption

To stop encryption a device shall send an LMP_encryption_mode_req PDU with the parameter encryption mode equal to 0 (no encryption). The other device responds with an LMP_accepted PDU or an LMP_not_accepted PDU (the procedure is described in [Sequence 31](#) in [section 4.2.5.1 on page 232](#)). If accepted, encryption shall be stopped by the master sending an LMP_stop_encryption_req PDU and the slave shall respond with an LMP_accepted PDU according to [Sequence 35](#).



Sequence 35: Stop of encryption.

Stopping encryption shall be performed in three steps, similar to the procedure for starting encryption.

1. Master is configured to transmit encrypted packets and to receive unencrypted packets.
2. Slave is configured to transmit and receive unencrypted packets.
3. Master is configured to transmit and receive unencrypted packets.

Between step 1 and step 2 master to slave transmission is possible. This is when an LMP_stop_encryption_req PDU is transmitted. Step 2 is triggered when the slave receives this message. Between step 2 and step 3 slave to master transmission is possible. This is when an LMP_accepted PDU is transmitted. Step 3 is triggered when the master receives this message.

4.2.5.5 Change encryption mode, key or random number

If the encryption key or encryption random number need to be changed or if the current link key needs to be changed according to the procedures in [Section 4.2.4 on page 230](#), encryption shall be stopped and re-started after completion, using the procedures in [Section 4.2.5 on page 232](#), subsections 3-4 for the new parameters to take effect.

4.2.6 Request supported encryption key size

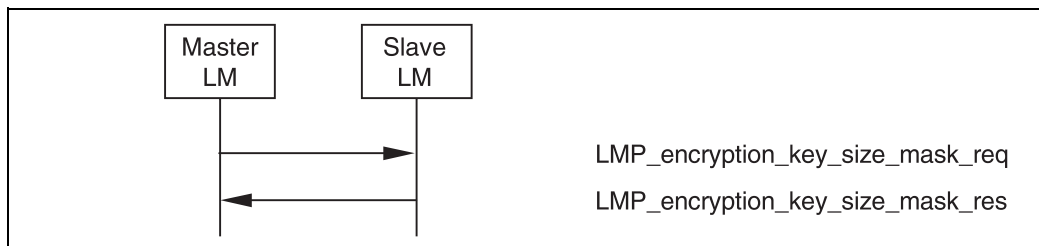
When broadcast encryption is supported via the LMP features mask, it is possible for the master to request a slave's supported encryption key sizes.

M/O	PDU	Contents
O(23)	LMP_encryption_key_size_mask_req	
O(23)	LMP_encryption_key_size_mask_res	key size mask

Table 4.16: PDUs used for encryption key size request

The master shall send an LMP_key_size_req PDU to the slave to obtain the slaves supported encryption key sizes.

The slave shall return a bit mask indicating all broadcast encryption key sizes supported. The least significant bit shall indicate support for a key size of 1, the next most significant bit shall indicate support for a key size of 2 and so on up to a key size of 16. In all cases a bit set to 1 shall indicate support for a key size; a bit set to 0 shall indicate that the key size is not supported.



Sequence 36: Request for supported encryption key sizes.

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

1	Security Overview	749
2	Random Number Generation	751
3	Key Management.....	753
3.1	Key Types	753
3.2	Key Generation and Initialization	755
3.2.1	Generation of initialization key, ...	756
3.2.2	Authentication.....	756
3.2.3	Generation of a unit key	756
3.2.4	Generation of a combination key.....	757
3.2.5	Generating the encryption key	758
3.2.6	Point-to-multipoint configuration.....	759
3.2.7	Modifying the link keys	760
3.2.8	Generating a master key	760
4	Encryption.....	763
4.1	Encryption Key Size Negotiation.....	764
4.2	Encryption of Broadcast Messages.....	764
4.3	Encryption Concept.....	765
4.4	Encryption Algorithm	766
4.4.1	The operation of the cipher	768
4.5	LFSR Initialization	769
4.6	Key Stream Sequence	772
5	Authentication	773
5.1	Repeated Attempts	775
6	The Authentication And Key-Generating Functions.....	777
6.1	The Authentication Function E1	777
6.2	The Functions Ar and A'r.....	779
6.2.1	The round computations.....	779
6.2.2	The substitution boxes “e” and “l”	779
6.2.3	Key scheduling	780
6.3	E2-Key Generation Function for Authentication.....	781
6.4	E3-Key Generation Function for Encryption.....	783
7	List of Figures.....	785
8	List of Tables	787



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 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 783](#)). 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, it will change frequently.

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

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, etc. Ideally, a true random generator based on some physical process with inherent randomness should be used as a seed. Examples of such processes are thermal noise from a semiconductor or resistor and the frequency instability of a free running oscillator. For practical reasons, a software based solution with a pseudo-random generator is probably preferable. In general, it is quite difficult to classify the randomness of a pseudo-random sequence. Within this specification, the requirements placed on the random numbers used are non-repeating and randomly generated.

The expression ‘non-repeating’ means that it shall be highly unlikely that the value will repeat itself within the lifetime of the authentication key. For example, a non-repeating value could be the output of a counter that is unlikely to repeat during the lifetime of the authentication key, or a date/time stamp.

The expression ‘randomly generated’ means that it shall not be possible to predict its value with a chance that is significantly larger than 0 (e.g., greater than $1/2^L$ for a key length of L bits).

The LM may use such a generator for various purposes; i.e. whenever a random number is needed (such as the RANDs, the unit keys, K_{init} , K_{master} , and random back-off or waiting intervals).



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 783](#).

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 760](#).

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 760](#).

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 759](#).

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 759](#).

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 got hold of 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_RANDOM. The principle is depicted in [Figure 6.4 on page 783](#). 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_RANDOM 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 775](#)).

The details of the E_{22} algorithm can be found in [Section 6.3 on page 781](#).

3.2.2 Authentication

The authentication procedure shall be carried out as described in [Section 5 on page 773](#). During each authentication, a new AU_RANDOM_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 758](#).

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 781](#). Once created, the unit key should be stored in non-volatile memory and very rarely changed. If after initialization

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 757](#), 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} .

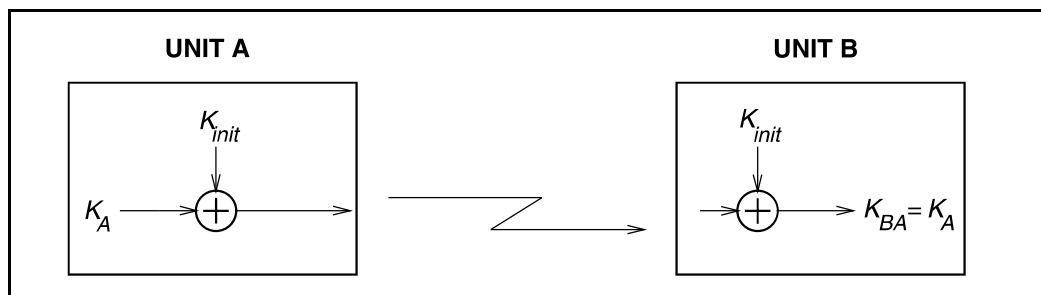


Figure 3.1: Generation of unit key. When the unit key has been exchanged, the initialization key is discarded in both devices.

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_{21} with the random number and their own BD_ADDRs , the two random numbers

$$LK_K_A = E_{21}(LK_RAND_A, BD_ADDR_A), \quad (EQ\ 1)$$

and

$$LK_K_B = E_{21}(LK_RAND_B, BD_ADDR_B), \quad (EQ\ 2)$$

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 757 and device B shall calculate (EQ 1) on page 757. 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 758.

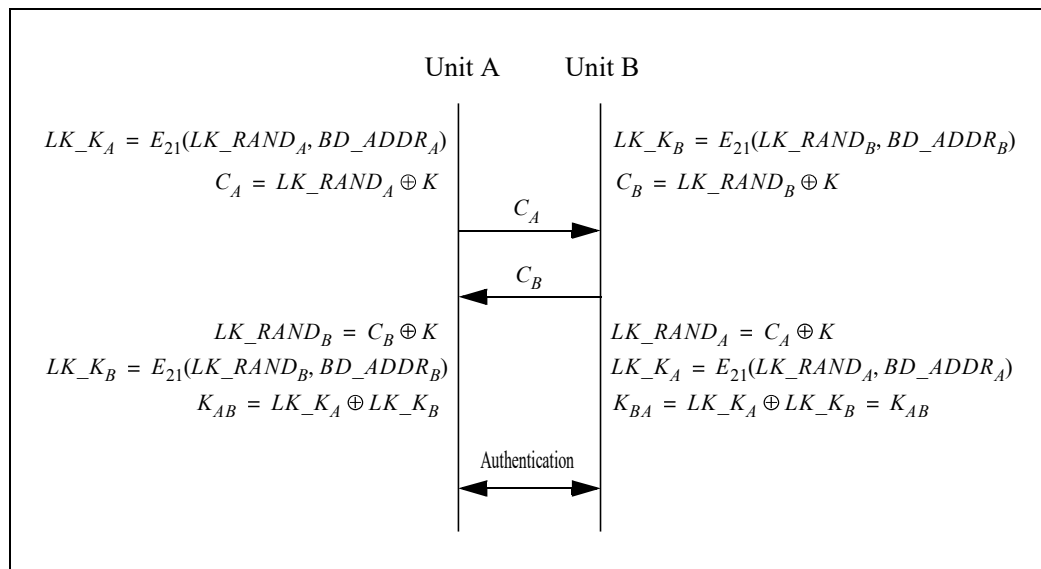


Figure 3.2: Generating a combination key. The old link key (K) is discarded after the exchange of a new combination key has succeeded

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:¹

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



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

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 encryption mode. The details of the key generating function E_3 can be found in [Section 6.4 on page 783](#).

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_{master} .

The transfer of necessary parameters shall be protected by the routine described in [Section 3.2.8 on page 760](#). 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_RANDOM 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 758](#), 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 759](#)), 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

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

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_{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_RANDOM, 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:

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

where the value of COF shall be derived from the master's BD_ADDR as specified by equation (EQ 3) on page 759. The details of the encryption key generating function are described in Section 6.4 on page 783. 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.

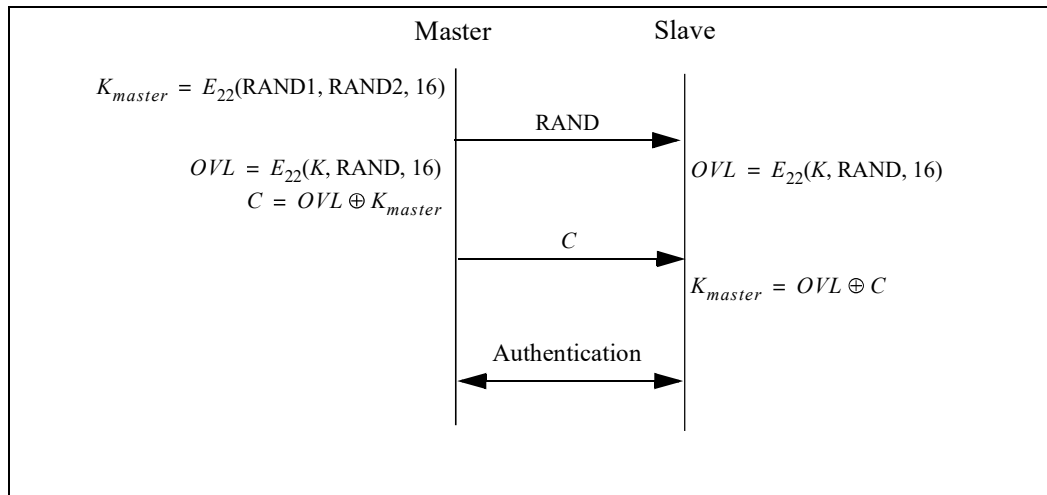


Figure 3.3: Master link key distribution and computation of the corresponding encryption key.



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 763](#).

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 this 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.

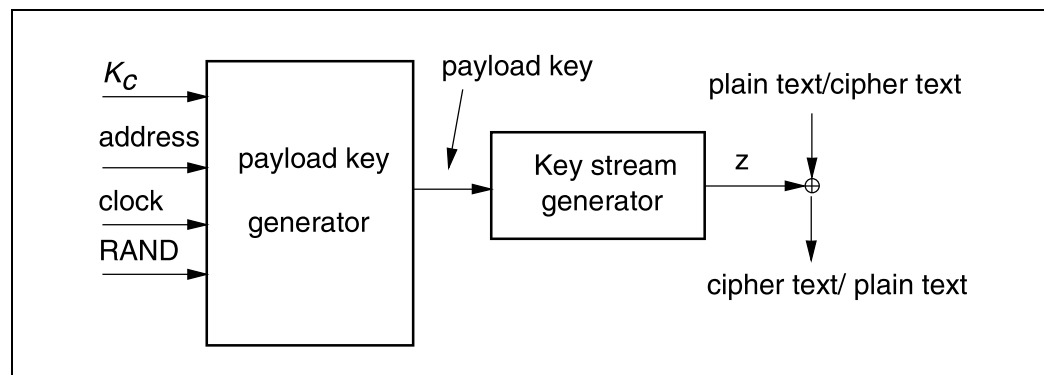


Figure 4.1: Stream ciphering for Bluetooth with E_0 .



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, $L_{sug}^{(M)}$, to the slave. Initially, the suggested value shall be set to $L_{max}^{(M)}$. If $L_{min}^{(S)} \leq L_{sug}^{(M)}$, 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, $L_{sug}^{(S)} < L_{sug}^{(M)}$, 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_{sug} and all smaller key lengths, or if $L_{sug} < L_{min}$ in one of the devices. In case of an abort link encryption can not 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_{master}
Encryption, K_{master}	Encryption, K_{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_0 uses the master Bluetooth device address (BD_ADDR), 26 bits of the master real-time clock (CLK₂₆₋₁) and the encryption key K_C as input, see [Figure 4.2 on page 766](#) (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, EN_RANDOM_A (see [Section 6.4 on page 783](#)). The random number shall be issued by the master before entering encryption mode. Note that EN_RANDOM_A is publicly known since it is transmitted as plain text over the air.

Within the E_0 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 769](#).

The real-time clock is incremented for each slot. The E_0 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 CLK₂₆₋₁ 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_0 generates a binary keystream, K_{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.

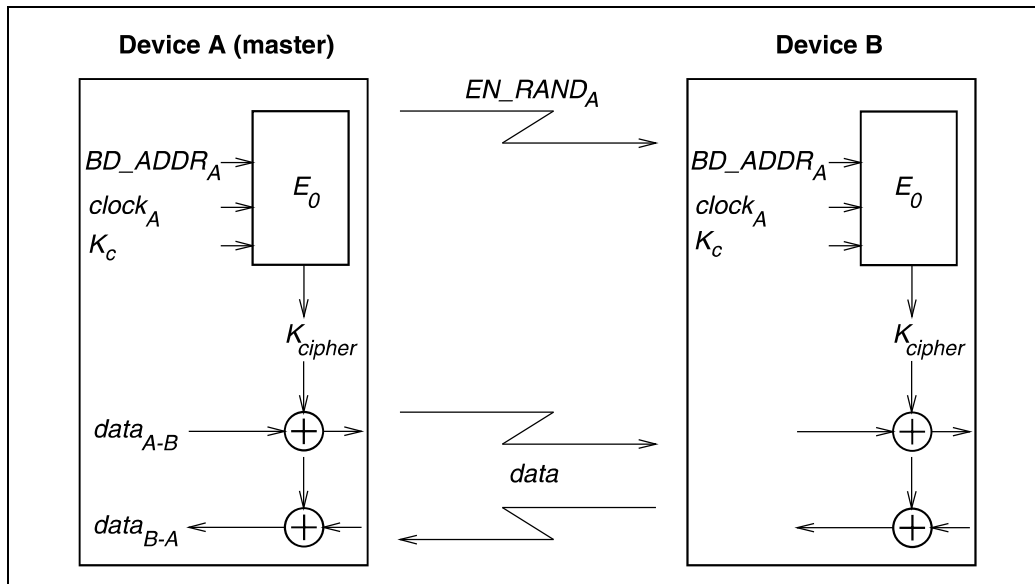


Figure 4.2: Functional description of the encryption procedure

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 767](#) 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 767](#). 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.

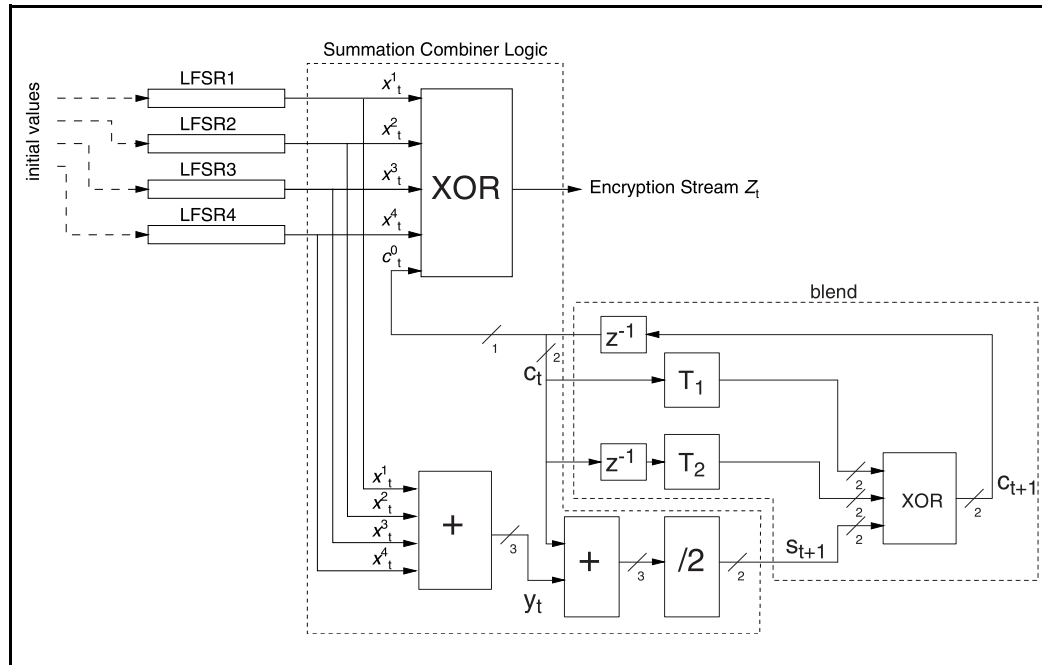


Figure 4.3: Concept of the encryption engine.

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^{th} symbol of LFSR _{i} . 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 $T_1[\cdot]$ and $T_2[\cdot]$ are two different linear bijections over GF(4). Suppose GF(4) is generated by the irreducible polynomial $x^2 + x + 1$, and let α be a zero of this polynomial in 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 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.

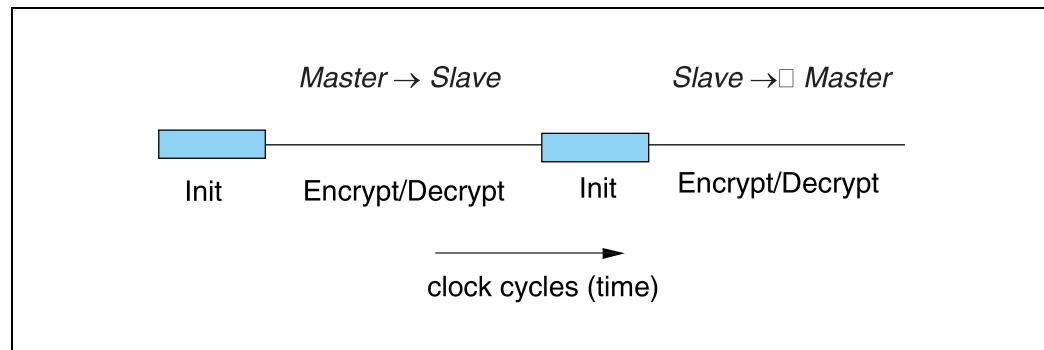
$$T_1: (x_1, x_0) \mapsto (x_1, x_0),$$

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

4.4.1 The operation of the cipher

[Figure 4.4 on page 768](#) 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 770.

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 771](#);
 - 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 LFSR_i , close the switch of this LFSR.
 - e) At $t = 39$ (when the switch of 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, LFSR_1 has effectively clocked 30 times with feedback closed, LFSR_2 has clocked 24 times, LFSR_3 has clocked 22 times, and 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 . ¹



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 .¹

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_ADDR₁₆ is entered, followed by BD_ADDR₁₇, etc. Furthermore, CL₀ corresponds to CLK₁, ..., CL₂₅ corresponds to CLK₂₆.

Note that the output symbols x_t^i , $i = 1, \dots, 4$ are taken from the positions 24, 24, 32, and 32 for LFSR₁, LFSR₂, LFSR₃, and LFSR₄, respectively (counting the leftmost position as number 1).

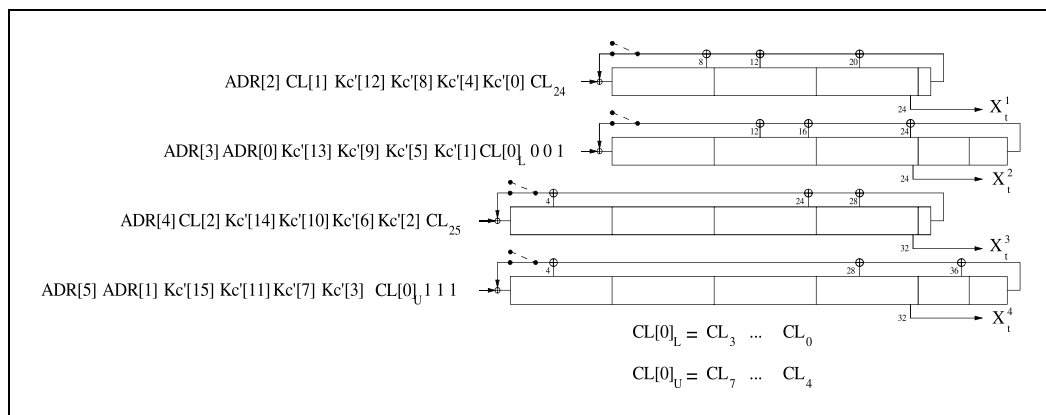


Figure 4.5: Arranging the input to the LFSRs.

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 LFSR₄.

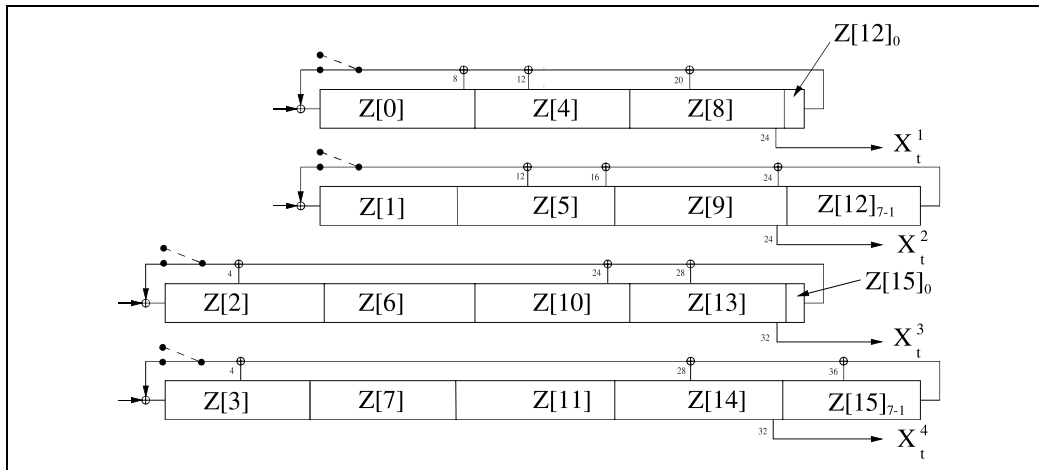


Figure 4.6: Distribution of the 128 last generated output symbols within the LFSRs.

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 649](#). 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_1 , and return the result $SRES$ to the verifier, see [Figure 5.1 on page 773](#). This figure also shows that the input to E_1 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 attack¹. The secret K shared by devices A and B is the current link key.

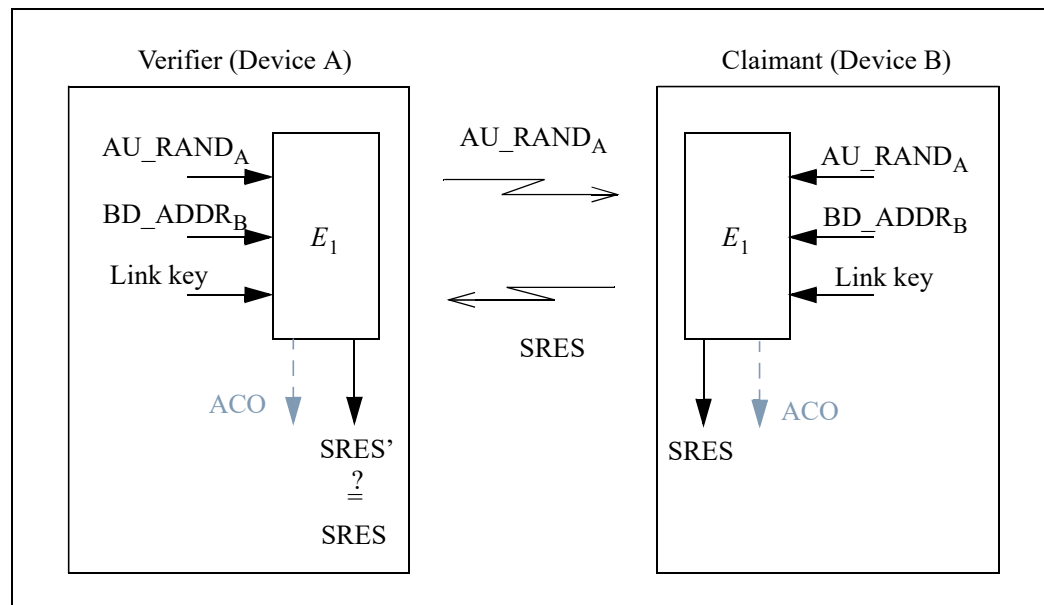


Figure 5.1: Challenge-response for the Bluetooth.

The challenge-response scheme for symmetric keys is depicted in [Figure 5.2 on page 774](#).

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.

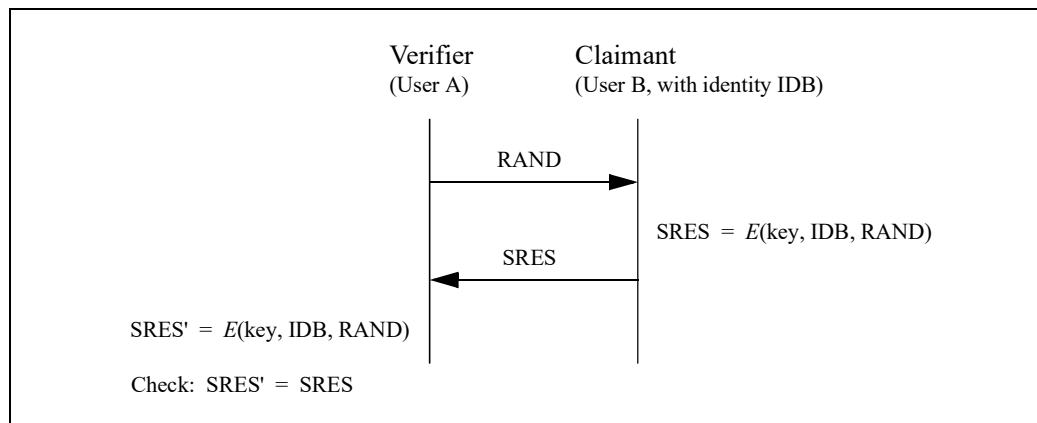


Figure 5.2: Challenge-response for symmetric key systems.

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 773](#), 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_1 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 with the same Bluetooth device address, 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 attempt¹. 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 make the system less vulnerable to denial-of-service attacks, the devices should keep a list of individual waiting intervals for each device it has established contact with. The size of this list may be restricted to only contain the N devices with which the most recent contacts have been made. The number N may vary for different devices depending on available memory size and user environment.

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_1

The authentication function E_1 is a computationally secure authentication code. E_1 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_1 is constructed using A_r as follows

$$E_1: \{0, 1\}^{128} \times \{0, 1\}^{128} \times \{0, 1\}^{48} \rightarrow \{0, 1\}^{32} \times \{0, 1\}^{96} \quad (EQ 12)$$

$$(K, \text{RAND}, \text{address}) \mapsto (\text{SRES}, \text{ACO}),$$

where $\text{SRES} = \text{Hash}(K, \text{RAND}, \text{address}, 6)[0, \dots, 3]$, where Hash is a keyed hash function defined as¹,

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

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

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_1 . The key $\tilde{K}$ for the second use of A_r (actually A'_r) is offset from K as follows²

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 778](#).

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 759](#) and [\(EQ 23\) on page 783](#). 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 777](#), i.e.

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

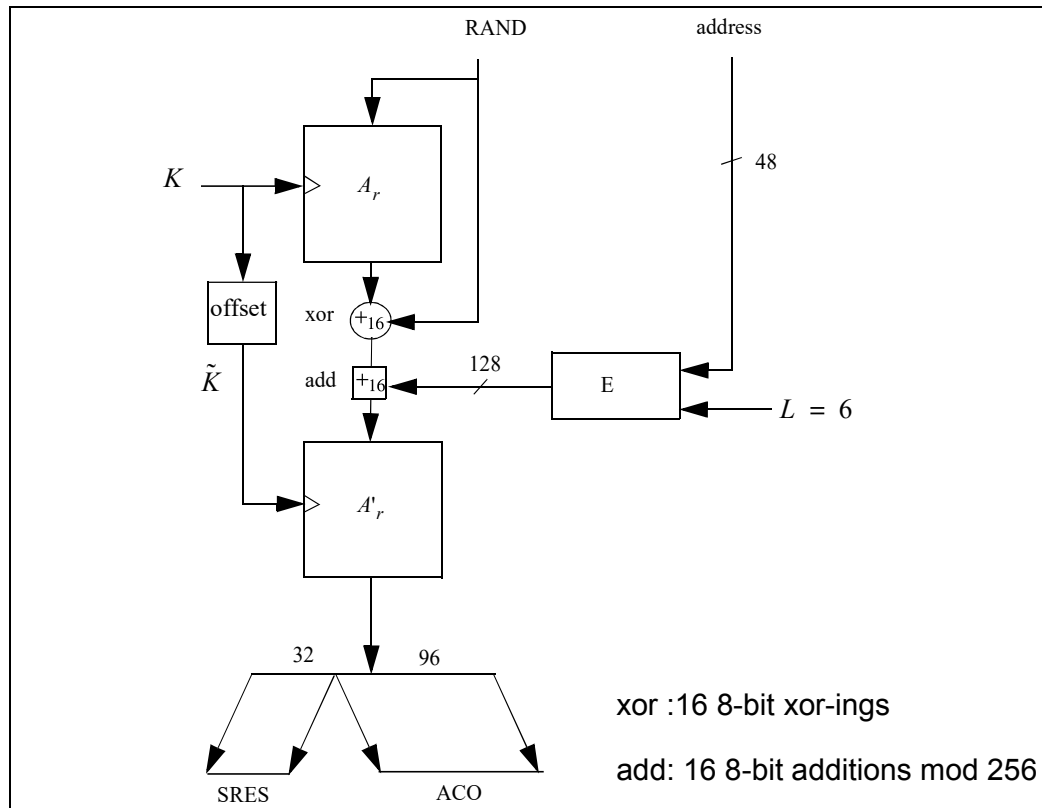


Figure 6.1: Flow of data for the computation of E_1 .



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 780](#) 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 780](#). 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 780](#) 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.

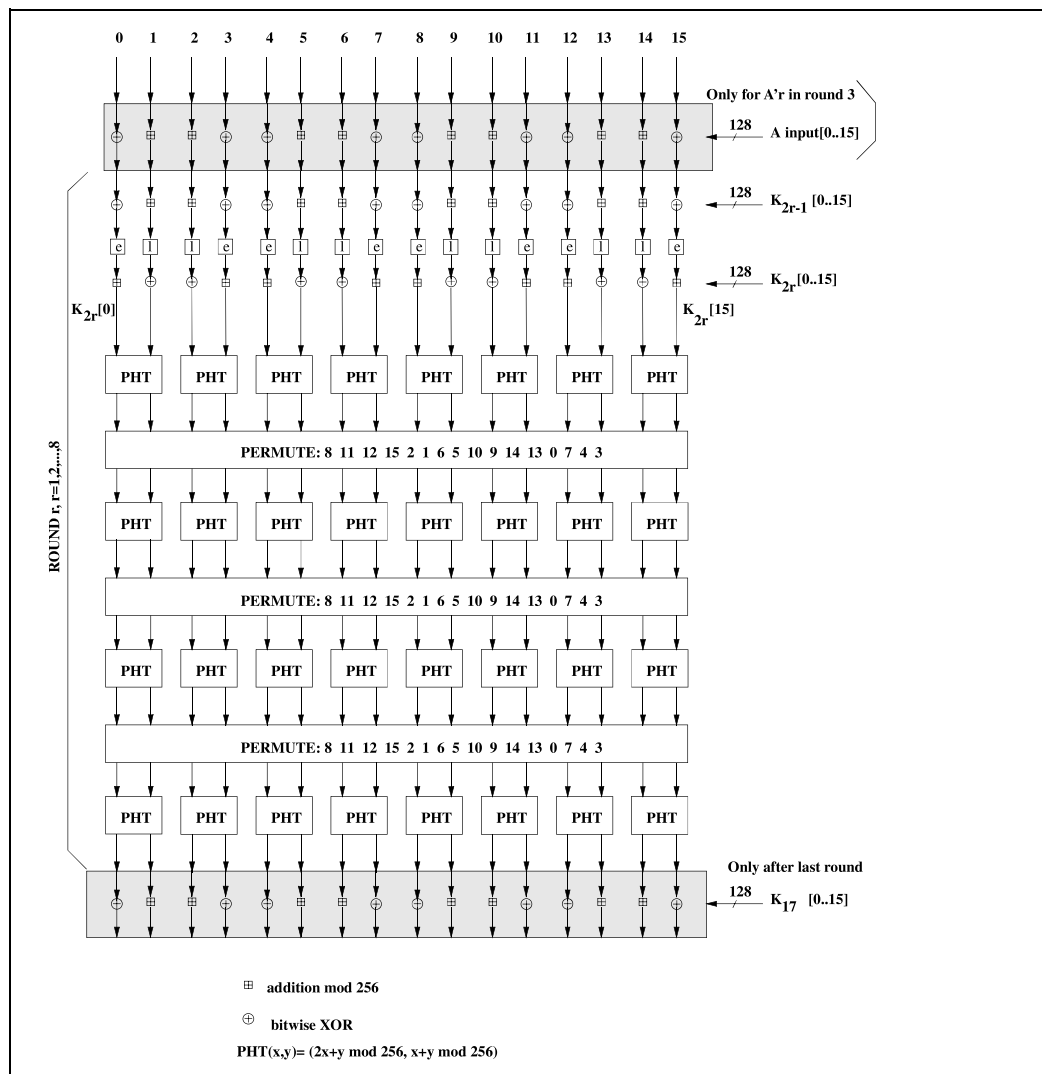
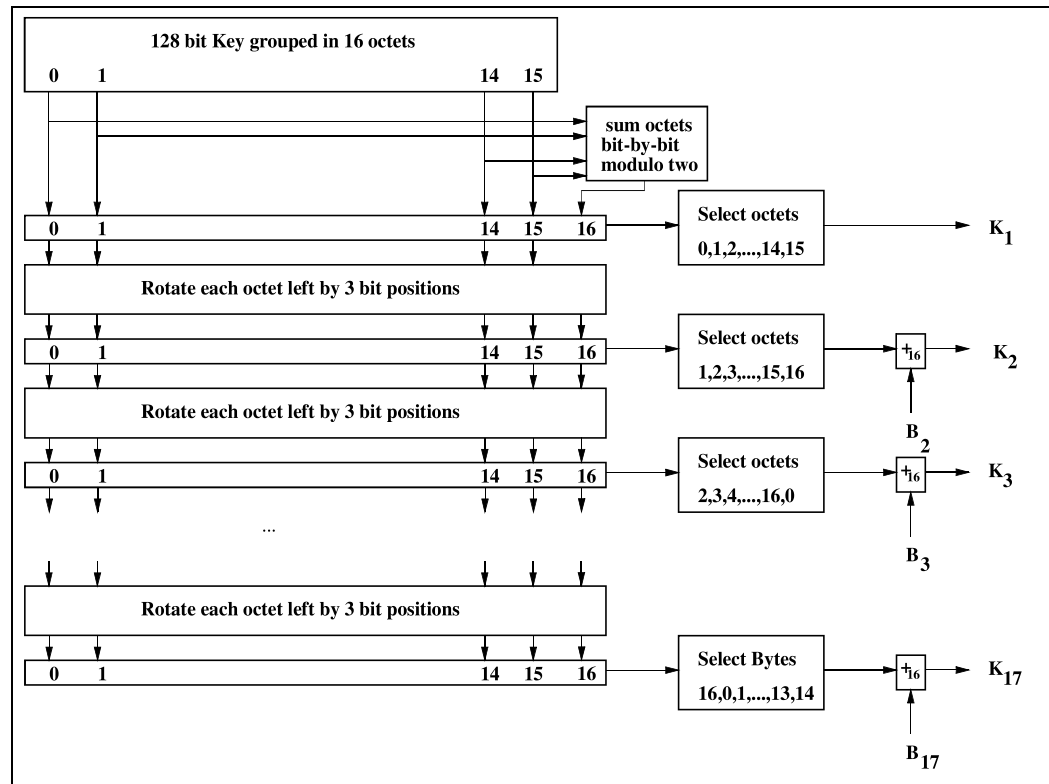


Figure 6.2: One round in A_r and A'_r . The permutation boxes show how input byte indices are mapped onto output byte indices. Thus, position 8 is mapped on position 0 (leftmost), position 11 is mapped on position 1, et cetera.

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 781 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:

$$B_p[i] = ((45^{(45^{17p+i+1} \bmod 257)} \bmod 257) \bmod 256), \text{ for } i = 0, \dots, 15. \quad (\text{EQ } 17)$$

Figure 6.3: Key scheduling in A_r .

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 783](#). 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 756](#) 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:

$$E_{21}: \{0, 1\}^{128} \times \{0, 1\}^{48} \rightarrow \{0, 1\}^{128} \quad (\text{EQ 18})$$

$$(\text{RAND}, \text{address}) \mapsto A'_r(X, Y)$$

where (for mode 1)

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

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

$$E_{22}: \{0, 1\}^{8L'} \times \{0, 1\}^{128} \times \{1, 2, \dots, 16\} \rightarrow \{0, 1\}^{128} \quad (\text{EQ 21})$$

$$(\text{PIN}', \text{RAND}, L') \mapsto A'_r(X, Y)$$

where

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

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

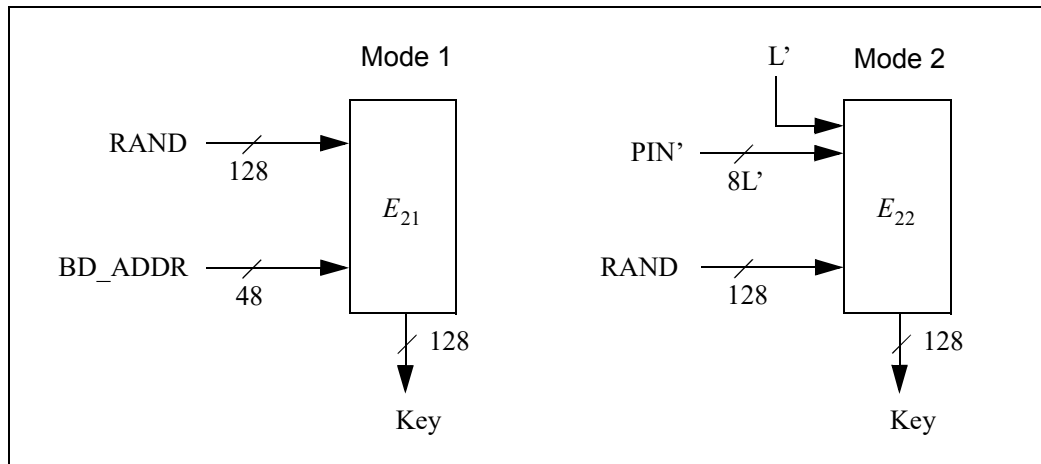


Figure 6.4: Key generating algorithm E_2 and its two modes. Mode 1 is used for unit and combination keys, while mode 2 is used for K_{init} and K_{master} .

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

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

$$(K, RAND, COF) \mapsto Hash(K, RAND, COF, 12)$$

where $Hash$ is the hash function as defined by (EQ 13) on page 777. 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 769. 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 759.

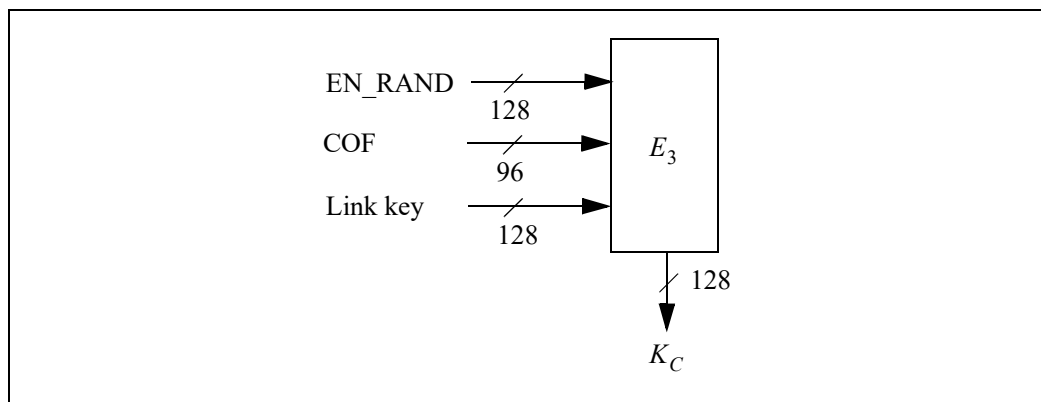


Figure 6.5: Generation of the encryption key.