BLUETOOTH DOC	Date / Year-Month-Day 2002-05-27	Approved Draft	Revision 1.0	Document No
Prepared Simon Morris	e-mail address simon.morris@csr.com			N.B. Confidential

RECOMMENDATIONS TO EARLY IMPLEMENTERS: ENCRYPTING BROADCAST TRANSMISSIONS IN BLUETOOTH™ PICONETS

Bluetooth SIG Security Expert Group

Abstract

This document describes the recommended approach to implementing broadcast encryption as specified in the Bluetooth 1.1 Specification.

Special Interest Group (SIG)

The following companies are represented in the Bluetooth Special Interest Group:

3Com Corporation
Ericsson Mobile Communications AB
IBM Corporation
Intel Corporation
Agere Systems, Inc.
Microsoft Corporation
Motorola, Inc.
Nokia Mobile Phones Ltd.
Toshiba Corporation

Revision History

Revision	Date	Comments
0.1	2001-04-12	First version.
0.2	2002-02-01	Updated after comments and Section 3 added.
0.3	2002-04-02	Updated after comments and examples added.
0.4	2002-18-03	Modified to be an Implementers Guide.
0.5	2002-02-05	Updated after final review before distribution.
0.6	2002-02-23	Updated title and introductory paragraph.

Contributors

Simon Morris	CSR
Jacob Jensen	Digianswer
Christian Gehrmann	Ericsson
Roland Hellfajer	Infineon

Disclaimer and Copyright Notice

THIS DOCUMENT IS PROVIDED "AS IS" WITH NO WARRANTIES WHATSOEVER, INCLUDING ANY WARRANTY OF MERCHANTABILITY, NON-INFRINGEMENT, FITNESS FOR ANY PARTICULAR PURPOSE, OR ANY WARRANTY OTHERWISE ARISING OUT OF ANY PROPOSAL, SPECIFICATION, OR SAMPLE. Any liability, including liability for infringement of any proprietary rights, relating to use of information in this document is disclaimed. No license, express or implied, by estoppel or otherwise, to any intellectual property rights are granted herein.

This document is for comment only and is subject to change without notice.

Contents

1 Introduction

This document by the Bluetooth Security Experts Group outlines a set of recommendations for encrypting broadcast transmissions in Bluetooth piconets. The recommendations aim to clarify relevant portions of the Bluetooth specification that may be interpreted in conflicting ways, possibly leading to interoperability problems when considering broadcasting encrypted data. Furthermore, two proposals on how to perform negotiations for the size of the broadcast encryption key are discussed. This document aims to encourage discussion on this topic that will influence the inclusion of any pertinent errata clarifications in the Bluetooth specification.

The document is organized as follows. First we give a short overview of broadcast encryption, how it is enabled, and how it is supposed to work. Furthermore, we discuss the deficiencies. In Section 2 we give recommendations on how to handle broadcast encryption. The recommendations are based on errata proposals and have been written with the goal of keeping the solution within the framework of the Bluetooth 1.1 Specification. Section 3 shows a state diagram describing the recommendations on how to implement broadcast encryption. Finally, in Section 4 we discuss two different broadcast encryption key size negotiation options.

1.1 Broadcast Encryption Overview

The Bluetooth Baseband Specification [1] describes the basic principles for broadcast encryption. The basis for broadcast encryption is the introduction of a *master key* in a piconet. The master must send the master key, K_{master}, to each slave in the piconet. This must be done *individually* between each slave and the master. The master key is generated by the master and distributed (securely) to all slaves in the piconet. Hence, since all slaves in the piconet share a common secret key, it is possible for the master to encrypt all broadcast traffic using an encryption key derived from this secret key and only the members in the piconet can decrypt the broadcast information.

There are two points where broadcast encryption can be enabled: during connection creation using the HCI command *HCI_Write_Encryption_Mode,* and after connection creation using the HCI command *HCI_Set_Connection_Encryption* [2].

1.1.1 Broadcast Encryption Enabled During Connection Creation

Here we assume that the HCI command *HCI_Write_Encryption_Mode* is issued prior to connection creation. Encryption is enabled using LMP commands during connection creation (see the LMP Specification [3]). The encryption mode parameter from the HCI command is used as the mode parameter in the *LMP_encryption_mode_req* PDU. There are three different encryption modes defined in the LMP Specification, Table 5.2 [3]:

0	No encryption (default)
1	Point-to-point encryption

2 Point-to-point and broadcast encryption

The initiating LM sends the encryption mode request PDU. The responding LM replies with either *LMP_accepted* or *LMP_not_accepted*. However, it is possible that K_{master} has not yet been negotiated, hence the encryption mode 2 would behave the same as encryption mode 1.

The encryption mode given in *HCI_Write_Encryption_Mode* is only used on connection creation and is not applicable after connection creation.

1.1.2 Broadcast Encryption Enabled After Connection Creation

Here we assume that the HCI command *HCI_Set_Connection_Encryption* is issued after the connection has been created. The natural interpretation of the specification is that the encryption mode used in the resulting LMP command *LMP_encryption_mode_req* (after the HCI command) depends on the current link key in use; i.e., if a semi-permanent link key is in use the encryption mode value should be 1: point- to-point encryption; if a master link key is in use the encryption mode should be 2: point-to-point and broadcast encryption.

1.2 Broadcast Encryption Deficiencies

1.2.1 Broadcast Encryption Enabled During Connection Creation

When broadcast encryption is enabled during connection creation there are interoperability issues with the current specification, as it can be interpreted in numerous ways, for example:

- Point-to-point encryption should be used during connection creation and it is up to the host to perform the broadcast encryption enabling after the link is created.
- The Link Manager will negotiate a semi-permanent link key during connection creation, change to the master link key, and negotiate encryption.
- The Link Manager will negotiate a semi-permanent link key during connection creation, negotiate encryption, then if broadcast encryption is accepted change to the master link key and start encryption.

Consequently, it is unclear which approach should be used. Section 2 describes the approach recommended by the Security Expert Group.

1.2.2 Broadcast Encryption Enabled After Connection Creation

Since the value of the encryption mode parameter in this case is assumed to be determined on the current link key, there is no need to have three possible encryption mode parameters. Hence, the usage of *both* encryption mode 1 and encryption mode 2 is questionable and the Bluetooth 1.1 Specification does not describe the expected usage of the parameter for this case. In Section 2 we give a recommendation of how to use the encryption mode parameter in order to remove this ambiguity.

1.2.3 Encryption Key Size Negotiation

The Bluetooth 1.1 Specification states in the Baseband Specification [1], Section 14.2.2.6 that:

"Note that the master must negotiate what encryption key length to use individually with each slave that wants to use the master key. In case the master already has negotiated with some of these slaves, it has knowledge of what sizes can be accepted. Clearly, there might be situations where the permitted key lengths of some units are incompatible. In that case, the master must have the limiting unit exc luded from the group."

What is the meaning of "the limiting unit"? Do we find the largest common key size acceptable by the most demanding device in the piconet or the highest number of links that can be included?

The meaning of "the unit should be excluded" needs clarification. Does this mean, for example, that the unit should be detached or encryption disabled on the offending device or left encrypted using an encryption key derived from the master link key but with a differing encryption key size?

In Section 4 we give a recommendation of how the encryption key negotiation can be implemented.

2 Broadcast Encryption Recommendation

2.1 Introduction

This section describes the various problems at the baseband, LM, and HCI levels and explains how the proposal will address the deficiencies.

The objective of this proposal is to ensure that current Bluetooth 1.1 Specification implementations will be unambiguous and will allow interoperability for devices using a master link key and hence broadcast encryption.

2.2 HCI_Write_Encryption_Mode

2.2.1 Deficiencies

Currently the specification is unclear if broadcast encryption during connection creation is allowed.

2.2.2 Proposed Solution

The proposal removes the need for broadcast encryption during connection creation by forbidding the use of broadcast encryption until a connection is complete at the HCI level; i.e., an *HCI_Connection_Complete* has been received by the host.

The *Encryption_Mode* parameter 0x02 (encryption for both point-to-point and broadcast) which can be set using the *HCI_Write_Encryption_Mode* command should not be used. To achieve interoperability on current 1.1 implementations, it is recommended that mode=0x02 is treated as mode=0x01, resulting in enabling of only point-to-point encryption during connection creation.

When the connection is complete the master host must issue the *HCI_Master_Link_Key* (temporary key) command to switch the link to the master link key. If subsequently new connections are made and encryption is enabled, they will have point-to-point encryption using an encryption key derived from the semi-permanent key until the host reissues *HCI_Master_Link_Key*.

2.3 Verifying the Master Link Key

2.3.1 Deficiencies

When changing to a temporary link key, if the mutual authentication failed, it is unclear what the action should be. Should the link revert to the semi-permanent link key or detach?

2.3.2 Proposed Solution

If mutual authentication of the temporary link key fails , the link should be detached and an *HCI_Disconnect* (Authentication Failure) event sent. This removes the problem of which status code should be reported to the master host in the *HCI_Master_Link_Key_Complete* event, as only a single *HCI_Master_Link_Key_Complete* is sent on the master regardless of the number of connected slaves.

2.4 Encryption Mode

2.4.1 Deficiencies

When encryption is requested using *LMP_encryption_mode_req* it is unclear what the encryption mode parameter should be set to; point-to-point or point-to-point and broadcast encryption.

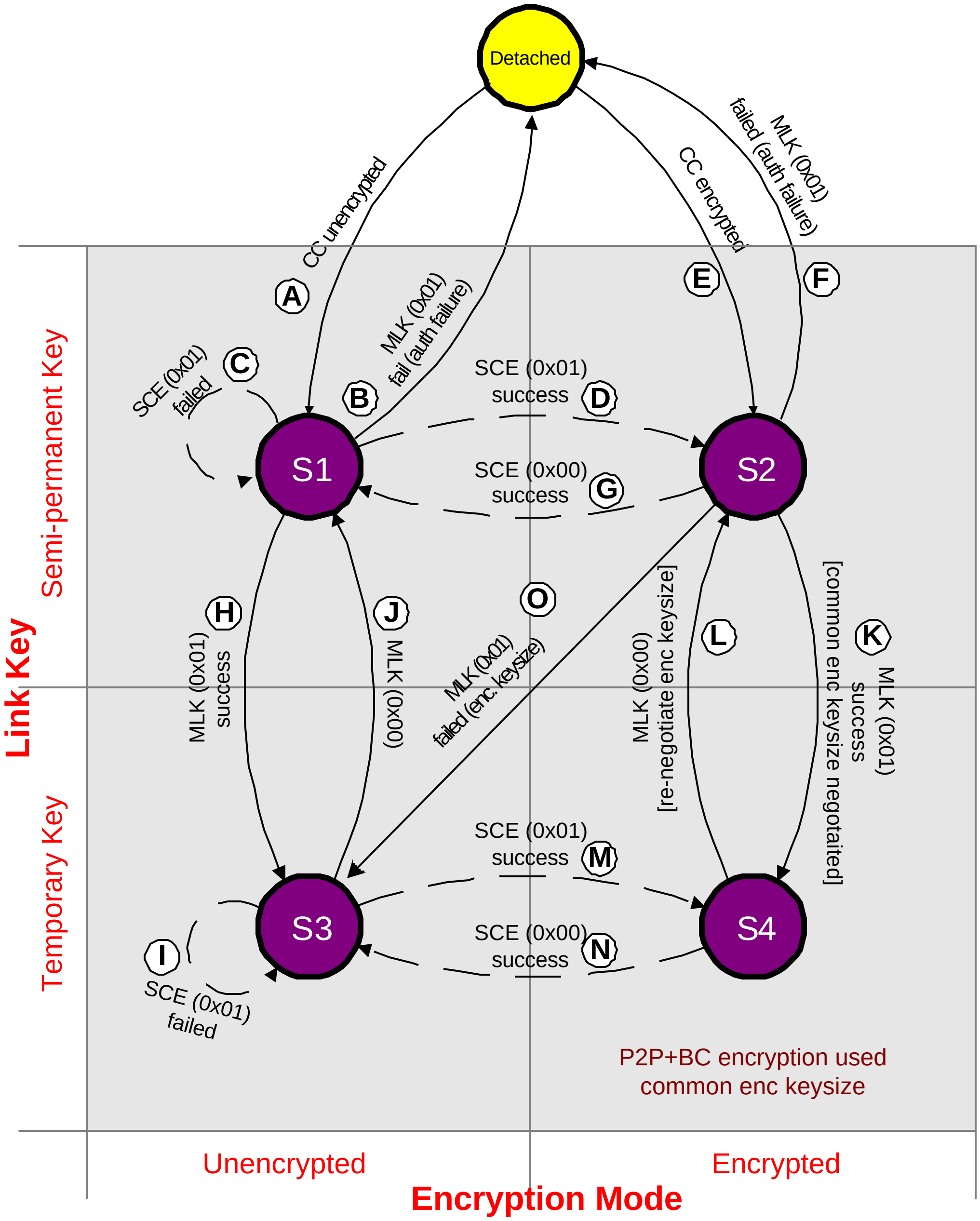
2.4.2 Proposed Solution

To remove the confusion about which mode should be sent in the PDU *LMP_Encryption_Mode_Req*, the allowed modes should be changed to *Encryption_On* and *Encryption_Off*. The mode in use is implied depending on the link key currently in use on the link. If a master link key is in use, then the point-to-point and broadcast encryption shall be used. If a semi-permanent link key is in use, point-to-point encryption shall be used.

3 Proposal Overview

3.1 Introduction

This section describes with the aid of a state diagram how the new broadcast encryption proposal will operate.



3.2 Description

The diagram above shows the possible states that a single instance of a connected link can be in with respect to encryption. There could be other links active in parallel. Below is an explanation of each of the states and events that cause transitions. The diagram assumes that a successful authenticated link can be established.

State	Description
Detached	Link does not exist.
S1	Encryption disabled. Semi-permanent link key in use.
S2	Encrypted point-to-point data. Semi-permanent link key in use.
S3	Encryption disabled. Temporary link key in use.
S4	Encrypted point-to-point and broadcast data. Temporary link key in use.

Event	Description
A	*HCI_Create_Connection* with encryption disabled has succeeded. The link has a semi-permanent link key.
B	*HCI_Master_Link_Key* (temporary key) has been requested by the host but mutual authentication has failed on the master link key hence the link is detached.
C	*HCI_Set_Connection_Encryption* (Enable) has been requested but the request failed for some reason.
D	*HCI_Set_Connection_Encryption* (Enable) has been requested and encryption key sizes have been agreed so the link is now encrypted. The link has a semi-permanent link key.
E	*HCI_Create_Connection* with encryption enabled has succeeded. With the new proposal we can only connect using point-to-point encryption. The link has a semi-permanent link key.
F	*HCI_Master_Link_Key* (temporary key) has been requested by the host but mutual authentication has failed on the master link key hence the link is detached.
G	*HCI_Set_Connection_Encryption* (Disable) has been requested and successfully disabled encryption.
H	*HCI_Master_Link_Key* (temporary key) has been requested by the host; as encryption is not enabled at this point no encryption change is seen. The piconet now has a master link key on the current connected links.
I	*HCI_Set_Connection_Encryption* (Enable) has been requested but the request failed for some reason.
J	*HCI_Master_Link_Key* (semi-permanent key) has been requested by the host; as encryption is not enabled at this point no encryption change is seen. The piconet now reverts back to the saved semi-permanent link key on all links. Authentication of the old key is not performed.
K	*HCI_Master_Link_Key* (temporary key) has been requested by the host; as encryption is enabled a common encryption key size needs to be agreed with all the slaves. See Section 4 for details. On completion of this , the piconet now has a common broadcast encrypted link.
L	*HCI_Master_Link_Key* (semi-permanent key) has been requested by the host. The piconet now reverts back to the saved link key on all links and encryption is turned off and back on, now using the original semi-permanent link key and ACO in the encryption key. Authentication of the old key is not performed.
M	*HCI_Set_Connection_Encryption* (Enable) has been requested. As a master link key is in use, the master attempts to turn encryption on using the agreed encryption key size; if there was no agreed key size the normal encryption key size negotiation is used. The link is now encrypted.

Event	Description
N	*HCI_Set_Connection_Encryption* (Disable) has been requested and successfully disables encryption. This link will now not be able to receive encrypted broadcast traffic.
O	*HCI_Master_Link_Key* (temporary key) has been requested by the host but a common encryption key size cannot be agreed for this link instance, so this instance is excluded from receiving encrypted broadcast traffic. See Section 4 for details.

3.3 Examples

In the next 3 sections typical scenarios are described in more detail. They show the command and event traffic at the HCI interface of the master device.

3.3.1 Simple Master Link Key Creation

This example shows a simple point-to-point connection in which the host wants to use point-to-point and broadcast encryption. The link is established with authentication and encryption (point-to-point) enabled. Then the host switches to master link key and back to semi-permanent link key.

State	Receive	HCI Commands and Events	Send
Detached			
		HCI_Write_Authentication_Enab le (0x01)	—>
	<—	HCI Command Complete (WAE)	
		HCI_Write_Encryption_Enable (0x01)	—>
	<—	HCI Command Complete (WEM)	
		HCI_Create_Connection (...)	—>
	<—	HCI Command Status (Create Connection)	
	<—	HCI Link Key Request Event (*bd_addr*)	
		HCI Link Key Request Reply (*bd_addr*, key)	—>
	<—	HCI Connection Complete (Point-to-Point Enc.)	
S2			
		HCI_Master_Link_Key (Temp. Key)	—>
	<—	HCI Command Status (Master link key)	
	<—	HCI Master Link Key Complete (Success)	
S4			
		...	
		HCI Master Link Key (Semi-perm anent Key)	—>
	<—	HCI Command Status (Master link key)	
	<—	HCI Master Link Key Complete (Success)	
S2			

3.3.2 Authentication of MLK Fails

This example shows what should happen if one of the links fails authentication of the master link key

State	Receive	HCI Commands and Events	Send
L1 – Detached			
		HCI_Write_Authentication_Enable (0x01)	—>
	<—	HCI Command Complete (WAE)	
		HCI_Write_Encryption_Enable (0x01)	—>
	<—	HCI Command Complete (WEM)	
		HCI_Create_Connection (...to slave 1)	—>
	<—	HCI Command Status (Create Connection)	
	<—	HCI Link Key Request Event(*bd_addr*)	
		HCI Link Key Request Reply (*bd_addr*, key)	—>
	<—	HCI Connection Complete (Point-to-Point Enc.)	
Link 1 - S2			
L2 -Detached			
		HCI_Write_Authentication_Enable (0x01)	—>
	<—	HCI Command Complete (WAE)	
		HCI_Write_Encryption_Enable (0x01)	—>
	<—	HCI Command Complete (WEM)	
		HCI_Create_Connection (...to slave 1)	—>
	<—	HCI Command Status (Create Connection)	
	<—	HCI Link Key Request Event (*bd_addr*)	
		HCI Link Key Request Reply (*bd_addr*, key)	—>
	<—	HCI Connection Complete (Point-to-Point Enc.)	
Link 2 - S2			
		HCI_Master_Link_Key (Temp. Key)	—>
	<—	HCI Command Status (Master link key)	
Link 1 – S4			
	<—	HCI Disconnection Complete (Success, Link 1)	
	<—	HCI Master Link Key Complete (Success)	
L2 - Detached			

3.3.3 One Device Already MLK, New Connection, and Go to MLK

This example shows one way to create two connections that use a common master link key and is using point-to-point and broadcast encryption. In this example the first connection is set up to use point-to-point and broadcast encryption before the second connection is initiated.

This example assumes that the two units share a common key size.

State	Receive	HCI Commands and Events	Send
L1+L2 Det.			
		HCI_Write_Authentication_Enable (0x01)	—>
	<—	HCI Command Complete (WAE)	
		HCI_Write_Encryption_Enable (0x01)	—>
	<—	HCI Command Complete (WEM)	
		HCI_Create_Connection (...to slave 1)	—>
	<—	HCI Command Status (Create Connection)	
	<—	HCI Link Key Request Event (*bd_addr*)	
		HCI Link Key Request Reply (*bd_addr*, key)	—>
	<—	HCI Connection Complete (Point-to-Point Enc.)	
Link 1 - S2			
		HCI_Master_Link_Key (Temp. Key)	—>
	<—	HCI Command Status (Master link key)	
	<—	HCI Master Link Key Complete (Success)	
Link 1 - S4			
		--- Time passes ---	
		HCI_Create_Connection (...to slave 2)	—>
	<—	HCI Command Status (Create Connection)	
	<—	HCI Connection Complete (Point-to-Point Enc.)	
Link 2 - S2			
		HCI_Master_Link_Key (Temp. Key)	—>
	<—	HCI Command Status (Master link key)	
	<—	HCI Master Link Key Complete (Success)	
Link 2 - S4			

4 Broadcast Encryption Key Size Negotiation

The following is a description of two negotiation methods for encryption key size negotiation that could be used when a system has a master link key and requires broadcast encryption.

4.1 Method 1: Key Size Negotiation

The meaning of "the unit should be excluded" is taken to be that encryption will be disabled on the excluded device and the current link key on the excluded device will be the master link key. Leaving encryption disabled ensures the host knows the device is excluded.

4.1.1 Negotiation

If we change to a temporary link key whilst encryption is on, we must stop and restart encryption to move onto the new encryption key. However, encryption key size negotiation may be needed; this can be achieved using the algorithm described below.

Some knowledge about encryption key size lengths can be made from previously encrypted links. The Bluetooth LMP Specification (Section 3.6.2) states that the master must initially send *key_size* = LmaxMaster (maximum key size it supports). If the slave cannot support this it sends back *key_size* = LmaxSlave (maximum key size it supports). Hence we know the largest possible key size allowed between these two devices on this connection. This algorithm assumes that the encryption key size range is contiguous.

Each device at manufacture has a preset minimum (Lmin) and maximum (Lmax) encryption key size that they will accept. If we have already negotiated encryption keys (as described above), we know what the maximum acceptable encryption key sizes are for each of the links. We do not know, however, the minimum encryption key size each device will accept, but we need to include as many devices in the broadcast encrypted piconet as possible. To achieve this we order the saved key sizes in ascending order from the minimum encryption key size to the maximum and apply the following algorithm:

(a.) Disable encryption on all links.

(b.) Take the lowest acceptable key size we know from the list of encryption key sizes already used (as described above) and attempt to turn encryption on to all devices.

(c.) If any devices refuse this key size, it means we must exclude all the slave devices that have a maximum encryption key size corresponding to the rejected size.

(d.) Disable encryption on all links.

(e.) Get the next largest key size we know about and attempt to turn encryption on to all non-excluded devices.

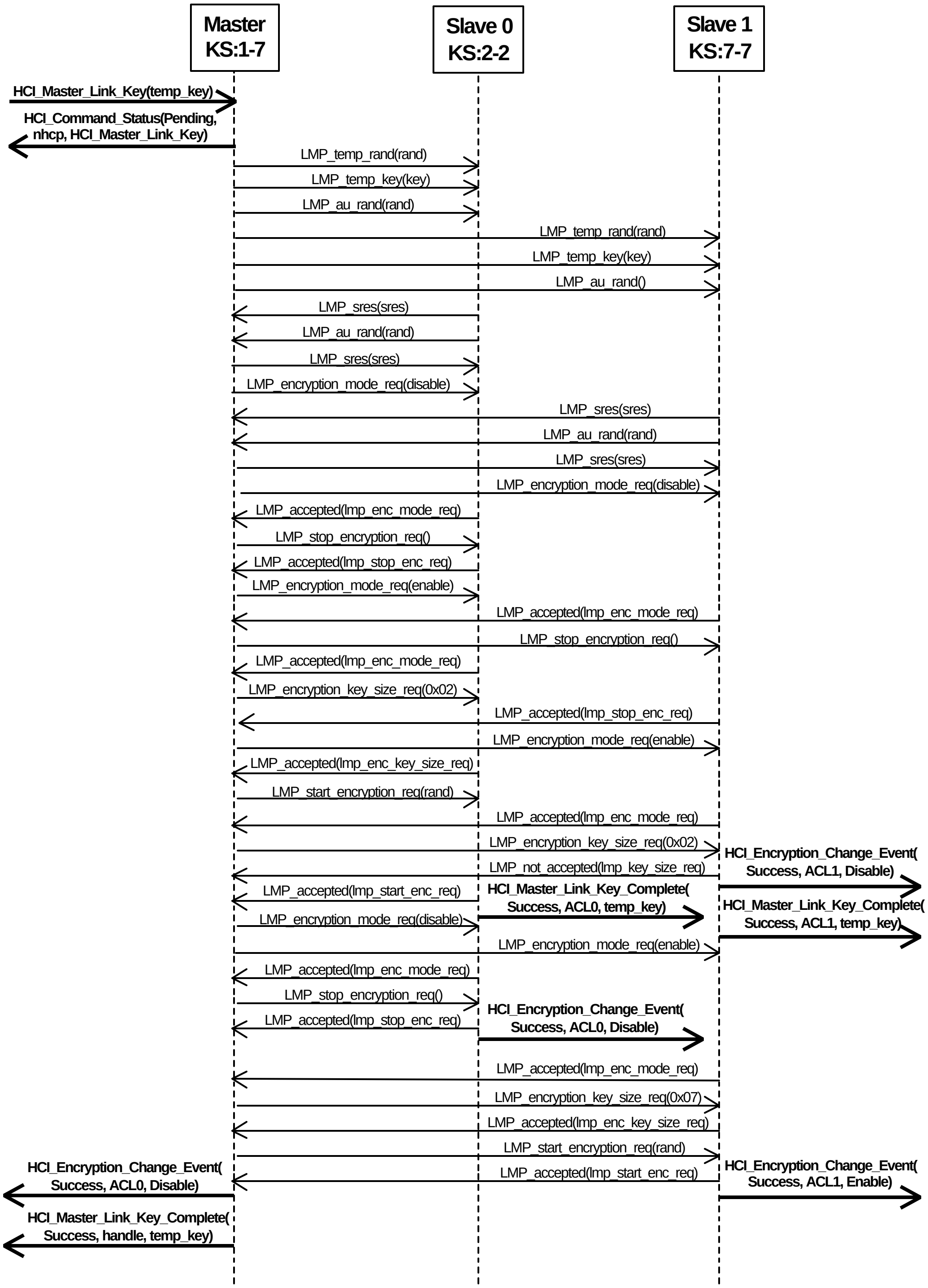
(f.) Repeat steps c, d, and e until we get agreement. The worst-case scenario is that we only have one link with encryption enabled.

An excluded device does not participate in any more key size negotiations as we know that it will not accept key lengths greater than what we have already tried. Hence when the key size negotiation has finished the excluded devices have encryption disabled. If the host wishes to enable encryption on this link it can send *HCI_Set_Connection_Encryption*, and the link will be encrypted using the master link key but with the excluded unit's negotiated key size, and point-to-point encryption but not broadcast encryption. (This means the link should not be parked.)

To prevent excessive HCI events being transmitted to the host, encryption mode change events should not be sent, as shown in the Bluetooth 1.1 Specification, Appendix IX (Message Sequence Charts), Section 4.4. If any devices are excluded, *HCI_Encryption_Change_Event* (Success, Handle, Disable) will be transmitted for each excluded device prior to the *HCI_Master_Link_Key_Complete_Event*. However, since the slave cannot know when encryption key size negotiation is complete, only the first encryption disable/enable pair will be hidden on the slave.

4.1.2 Example

The master has an allowable key size range of 1-7; slave 0 (acl0) only allows a key size 2; slave 1 (acl1) only allows a key size of 7. Both links are encrypted, hence the master has a record of the key sizes allowed of m->s0 = 2, m ->s1=7.


Master
KS:1-7
Slave 0
KS:2-2
Slave 1
KS:7-7
HCI_Master_Link_Key(temp_key)
HCI_Command_Status(Pending, nhcp, HCI_Master_Link_Key)
LMP_temp_rand(rand)
LMP_temp_key(key)
LMP_au_rand(rand)
LMP_temp_rand(rand)
LMP_temp_key(key)
LMP_au_rand()
LMP_sres(sres)
LMP_au_rand(rand)
LMP_sres(sres)
LMP_encryption_mode_req(disable)
LMP_sres(sres)
LMP_au_rand(rand)
LMP_sres(sres)
LMP_encryption_mode_req(disable)
LMP_accepted(lmp_enc_mode_req)
LMP_stop_encryption_req()
LMP_accepted(lmp_stop_enc_req)
LMP_encryption_mode_req(enable)
LMP_accepted(lmp_enc_mode_req)
LMP_stop_encryption_req()
LMP_accepted(lmp_enc_mode_req)
LMP_encryption_key_size_req(0x02)
LMP_accepted(lmp_stop_enc_req)
LMP_encryption_mode_req(enable)
LMP_accepted(lmp_enc_key_size_req)
LMP_start_encryption_req(rand)
LMP_accepted(lmp_enc_mode_req)
LMP_encryption_key_size_req(0x02)
HCI_Encryption_Change_Event(Success, ACL1, Disable)
LMP_not_accepted(lmp_key_size_req)
LMP_accepted(lmp_start_enc_req)
HCI_Master_Link_Key_Complete(Success, ACL0, temp_key)
HCI_Master_Link_Key_Complete(Success, ACL1, temp_key)
LMP_encryption_mode_req(disable)
LMP_encryption_mode_req(enable)
LMP_accepted(lmp_enc_mode_req)
LMP_stop_encryption_req()
HCI_Encryption_Change_Event(Success, ACL0, Disable)
LMP_accepted(lmp_stop_enc_req)
LMP_accepted(lmp_enc_mode_req)
LMP_encryption_key_size_req(0x07)
LMP_accepted(lmp_enc_key_size_req)
LMP_start_encryption_req(rand)
HCI_Encryption_Change_Event(Success, ACL0, Disable)
LMP_accepted(lmp_start_enc_req)
HCI_Encryption_Change_Event(Success, ACL1, Enable)
HCI_Master_Link_Key_Complete(Success, handle, temp_key)

4.1.3 Subsequent Encryption Enabling

If an active device with encryption disabled was switched to the master link key, it may decide to enable encryption at a later time using *HCI_Set_Connection_Encryption*.

If broadcast encryption is currently enabled, key negotiation must first be tried with the encryption key size currently in use. If this fails , renegotiation of the encryption key size will need to be performed. Renegotiation can be achieved by the master host reverting back to the semi-permanent link key using *HCI_Master_Link_Key* (semi-permanent key), enabling encryption using *HCI_Set_Connection_Encryption* followed by requesting a temporary link key again using *HCI_Master_Link_Key* (temporary key).

4.1.4 Leaving the Piconet

If a device leaves the piconet or disables encryption, the current encryption key size should stay in force until a new device is added.

4.1.5 Changing Back to a Semi-Permanent Key

When the link is returned to a semi-permanent link key, any excluded devices should have encryption re-enabled, unless explicitly disabled by the master by sending *HCI_Set_Connection_Encryption* (off).

4.1.6 New Connections

If a new device joins the piconet, it will not have a master link key. To change to a master link key, *HCI_Master_Link_Key* (temporary key) must be reissued by the master. This will force renegotiation of encryption key sizes if required.

4.1.7 Recommended Use

It is recommended that to achieve the best fit, the following procedure should be used when enabling broadcast encryption or updating broadcast encryption when new devices are added:

1. All links should revert back to using a semi-permanent link key if a temporary key was in use.
2. Point-to-point encryption is enabled on all devices wishing to participate in the broadcast encryption piconet.
3. *HCI_Master_Link_Key* is issued on the master.
4. Encryption key size negotiation is performed automatically from a known starting position.

4.2 Method 2: Fixed Key Size

An alternative method would be to impose a fixed key size when broadcast encryption is required. This key size could be, for example, fixed at 5 to give a reasonable amount of security. Devices not supporting this key size would be excluded from participating in broadcast encryption.

5 References

[1] Bluetooth SIG, Specification of the Bluetooth System, Core
Part B: Baseband Specification
Version 1.1, 22 February 2001, http://www.bluetooth.com/

[2] Bluetooth SIG, Specification of the Bluetooth System, Core
Part H-1: Host Controller Functional Specification
Version 1.1, 22 February 2001, http://www.bluetooth.com/

[3] Bluetooth SIG, Specification of the Bluetooth System, Core
Part C: Link Manager Protocol
Version 1.1, 22 February 2001, http://www.bluetooth.com/