

---

---

# Bluetooth

## *Part 10: Introduction to Wireless Security*

**Kjell Jørgen Hole**

UiB



Last updated 02.03.08

Mail: [Kjell.Hole@ii.uib.no](mailto:Kjell.Hole@ii.uib.no)

URL: [www.kjhole.com](http://www.kjhole.com)

## Outline

[KJhole.com](http://KJhole.com)

- Need for security
- Security toolbox: authentication, authorization, and encryption
- Security modes and architecture
- Security procedures
- Key generation and encryption algorithms

## Need for Security

KJhole.com

- Anyone with a Bluetooth-enabled device can potentially connect to your Bluetooth device, gaining access to data without your knowledge and permission
- As an example, insufficient security may give other users unrestricted access to the mobile phone network via your Bluetooth-enabled phone
- Security needs depend on the application being developed. Ultimately, the decision on how to implement security is up to the application developer

10.3

## Security Toolbox

KJhole.com

- The Bluetooth “security toolbox” is based on the three components:
  - Authentication:** used to verify the identity of a device
  - Authorization:** determines if a device is to be granted access to specific services offered by another device
  - Encryption:** protects data by encoding it prior to transmission
- No support for integrity!

10.4

**Pairing** Procedure involving exchanging link management packets to establish a temporary key, called an **initialization key** ( $K_{init}$ ), for use between two Bluetooth devices wishing to communicate for the *first* time

- The pairing procedure requires that an identical *Personal Identification Number* (PIN)\* be made available to both devices
- An application must ask the user for the PIN and deliver it to the Bluetooth stack
- If both devices have a fixed PIN, the devices cannot be paired

\*Also referred to as a pass-key

10.5

- During authentication a device determines whether or not it shares a common **authentication key** with another device
  - if the two devices are new to one another, the pairing procedure is needed to create the initialization key  $K_{init}$
  - this initialization key is then used to create a *semi-permanent authentication key* ( $K_{AB}$ ) which is authenticated
- Authentication can be triggered via HCI (Host Controller Interface) commands at any time

10.6

**Bonding** Refers to the entire process of link-creating, pairing, creation of the semi-permanent authentication key  $K_{AB}$ , and authentication

- Once devices are bonded, pairing does not have to be done again and authentication can proceed (using  $K_{AB}$ ) without the need for PIN entry
- If a device is requested to bond with another device that it already possesses an authentication key for, this key is erased. Pairing is then initiated, establishing another authentication key  $K_{AB}$

10.7

- Authorization is needed before a device is given permission to access a particular service
- Authorization requires that
  - requesting device is authenticated
  - service being requested is reported to device providing service
  - device determines whether or not to permit access to service
- An Man-Machine Interface (MMI) is often used to grant *Trust*

10.8

**Trust** Attribute that links authorization permission to a particular device

- If a device is marked as Trusted, then the authorization process can complete successfully without user interaction
- Trust can be granted both temporarily and permanently
- Permanent Trust is usually granted during the initial authorization via the MMI

10.9

- It is not possible to prevent the interception of data that is transmitted wirelessly
- Encryption relies upon a special **encryption key** ( $K_c$ ) generated from the stored authentication key  $K_{AB}$
- Authentication keys changes less frequently than encryption keys
- To underline the importance of the authentication key to a specific Bluetooth link, it is often referred to as the *link key*
  - we will *not* use the term link key

10.10

## Security Modes

KJhole.com

- The Generic Access Profile (GAP) defines three security modes:

**Security Mode 1** *No security*—Devices will never initiate any security procedure. Must respond to authentication challenge, may be able to turn on encryption

**Security Mode 2** *Service level-enforced security*—Channel or service using an L2CAP connection decides whether or not security is required

**Security Mode 3** *Link level-enforced security*—Device initiates security procedures before it sends an *LMP\_setup\_complete* message. If security measures fail, the connection will not be set up

10.11

## More on Security Modes

KJhole.com

- Note that in Mode 2 the device initiates security procedures **after** the channel is established (at the higher layers), while in Mode 3 the device initiates security procedures **before** the channel is established (at the lower layers)
- The Bluetooth security white paper defines a *security architecture* which may be used to implement Mode 2 service level-enforced security
  - risk of faulty implementation of security policy

10.12

## Device Security Levels

KJhole.com

- The security white paper splits devices into three security categories:

**Trusted devices** Paired or bonded devices which are marked in a database as trusted, and can be given unrestricted access to all services

**Known untrusted devices** Devices which have been paired or bonded, but are not marked in a database as trusted; access to services may be restricted

**Unknown devices** No security information is stored, devices are untrusted, and access to services may be restricted

10.13

## Service Security Levels

KJhole.com

- The security white paper also splits services into three security categories:

**Open services** Any device may access these; there are no security requirements

**Authentication-only services** Any device which can go through authentication may access these

**Authentication and authorization services** Only trusted devices may access these

10.14

## Security Architecture: Databases and Security Manager

---

- Since there exist trusted devices and different levels of authorization, **databases** are needed to hold device and service information
- Different protocols will access the information in these databases according to the implemented profile
- To allow uniform access to the databases by all protocols, a **security manager** handles security transactions with the different protocols

10.15

## Security Manager (1)

---

KJhole.com

- Applications and protocols wishing to use security features register with the security manager
- Security policies are enforced by exchanging queries with the security manager
- When security is triggered at the L2CAP layer (Mode 2 security), it is possible to block access to the above layers (SDP, TCS, RFCOMM, OBEX, ...)

10.16

## Security Manager (2)

KJhole.com

- The security manager must provide an application interface to:
  - configure security
  - request PIN entry
  - query the user for an authorization response
  - respond to the LM (Link Manager) with PIN information or an authentication key

10.17

## Databases

KJhole.com

**Service database** holds the security configuration information as provided by the application software

**Device database** stores information regarding past sessions with other Bluetooth nodes, allowing quick connections to be established without having to traverse the security barrier again

10.18

## Mode 2 Security Operation (1)

KJhole.com

Assume that L2CAP is the security trigger that invokes the security manager. Here is what happens when authentication, authorization, and encryption are required:

**Authentication 1** Security manager commands the host controller to authenticate the other device

**Authentication 2** Host controller responds, asking for an authentication key (if one exists)

10.19

## Mode 2 Security Operation (2)

KJhole.com

**Authentication 3** The device database is checked by the security manager for an authentication key associated with the address of the device being authenticated (assume no key exists yet)

**Authentication 4** The host responds with “no key”

**Authentication 5** The host controller makes a request for a PIN and the security manager asks the application for a PIN (either through a UI or from memory)

10.20

## Mode 2 Security Operation (3)

KJhole.com

**Authentication 6** The PIN is returned to the host controller and a temporary initialization key  $K_{init}$  is created

**Authentication 7** A semi-permanent authentication key  $K_{AB}$  is created and shared between devices

**Authentication 8** Authentication proceeds using this semi-permanent authentication key in a *challenge-response scheme*

**Authentication 9** The authentication key is sent to the host for storage in the device database for future reference

10.21

## Mode 2 Security Operation (4)

KJhole.com

**Authorization 1** The security manager examines the device database to see if the device is Trusted (assume it isn't yet)

**Authorization 2** The security manager presents the name of the device attempting to make a connection, and the service it wants to access to the application software. The application must respond back to the security manager if this connection is to

1. be authorized, and
2. if this device is to be Trusted

10.22

## Mode 2 Security Operation (5)

KJhole.com

**Authorization 3** The Trust attribute is entered into the device database by the security manager and the connection is permitted to proceed in establishing itself

**Encryption 1** The security manager then commands the host controller to invoke encryption, which it does

10.23

## Implementation: Security at Many Levels

- Baseband security algorithms specified in separate part of standard (not part of baseband specification)
- LM specification includes link level procedures for configuring security
- HCI specification details how a host controls security and how security-related events are reported by a controller to its host

10.24

## Security at Baseband

KJhole.com

Four entities are used for maintaining security at baseband:

<i>Entity</i>	<i>Size in bits</i>
RAND (random number)	128
BD_ADDR	48
authentication key	128
encryption key, configurable length (byte-wise)	8–128

**Remark:** The BD\_ADDR is the 48-bit IEEE address which is unique for each Bluetooth device. It is publicly known

10.25

## Random Number Generation

KJhole.com

- Each Bluetooth device has a pseudo-random number generator used for:
  - generating initialization keys  $K_{init}$  and authentication keys  $K_{AB}$ , as well as encryption keys  $K_c$
  - challenge-response scheme for authentication
- The random number generator must pass 16 statistical tests listed in the Bluetooth standard (v2.1)

10.26

## Key Generation with SAFER+ Algorithm

- The algorithms used to generate authentication and encryption keys are modified versions of a 128-bit block cipher algorithm, called *SAFER+* (Secure And Fast Encryption Routine)
- *SAFER+* was one of the candidates to the new *Advanced Encryption Standard* (AES), successor to the *Data Encryption Standard* (DES)

10.27

## More on SAFER+

[KJhole.com](http://KJhole.com)

- The U.S. *National Institute of Standards and Technology* (NIST) found that *SAFER+* has
  - a good security margin with only some minor security gaps that do not affect the 128-bit version used in Bluetooth
  - relatively slow speed , especially for 32 and 64 bit microprocessor-based implementations
- The algorithm's regular structure and byte orientation makes it suitable for implementation in silicon and small footprint microprocessors (i.e., 8 bit)

10.28

## More on Authentication Keys

KJhole.com

- Master and Slave **must** share the same secret authentication key to use encryption
- The secret authentication key is **never** transmitted in plaintext
- Authentication keys have different lifetimes:
  - used for several subsequent connections
  - lifetime limited by the lifetime of single connection

10.29

## Types of Authentication Keys

KJhole.com

- Three authentication (link) keys exist:
  - **Master key:**  $K_{\text{master}}$ , temporary key for point to multipoint communications
  - **Combination key:**  $K_{AB}$ , based on information from two devices A and B
  - **Initialization key:**  $K_{\text{init}}$ , temporary key used during initialization of device when combination key does not yet exist
- The combination key is a *semipermanent key* while the two other keys are *temporary keys*

10.30

- Encryption key  $K_c$  is loaded into four Linear Feedback Shift Registers (LFSRs) together with the BD address, clock bits, and another 128 bit random number (publicly known), all obtained from Master
- Output of LFSRs (payload key) is combined by a finite state machine, called the *Summation Combiner*, to produce a cipher stream which is XOR'd with the data stream
- LFSRs + Summation Combiner = *Stream Cipher*

10.31

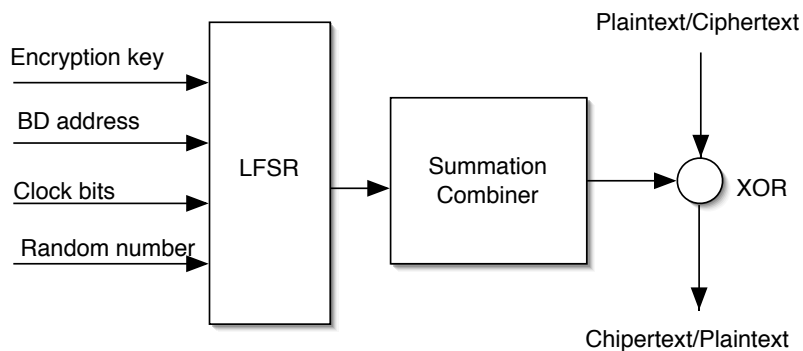


Figure 10-1 Stream cipher usage

10.32

## Stream Cipher

KJhole.com

- Operation of stream cipher is referred to as the “ $E0$ ” algorithm
- Since  $E0$  is reinitialized at the start of each new packet, a new cipher stream will be generated for each packet
- The payloads of SCO and ACL packets are encrypted over a link

!!! Note that SAFER+ is only used to generate keys for the  $E0$  algorithm

10.33

## Key Generation for Two Devices

KJhole.com

- Any two devices wanting to use authentication and encryption must execute the following four steps:

**Step 1** Generation of an initialization key  $K_{init}$

**Step 2** Generation and exchange of authentication key  $K_{AB}$

**Step 3** Authentication (challenge-response scheme)

**Step 4** Generation of encryption key  $K_c$  in each device

10.34

## Step 1: Usage of Initialization Key

- The 128 bit initialization key  $K_{\text{init}}$  is used for a single session and is created each time the device is initialized
  - the key is generated using the so-called  $E_{22}$  implementation of SAFER+
- The initialization key used for key exchange during the generation of an authentication key (step 2)
  - when the devices have performed the authentication key exchange, the initialization key is discarded

10.35

## Step 1: Generation of Initialization Key

- One device chooses a random number and transmits it to the other device
- Using  $E_{22}$ , both devices compute  $K_{\text{init}}$  based on
  - shared PIN
  - BD\_ADDR of device that received the random number
  - the random number itself

10.36

## Step 2: Generation of Authentication Key

1. Devices A and B choose random numbers,  $R_A$  and  $R_B$
2. Using algorithm  $E_{21}$ , device A (B) computes  $K_A$  ( $K_B$ ) as a function of  $R_A$  ( $R_B$ ) and the BD\_ADDR
3. Device A (B) sends  $R_A$  ( $R_B$ ) in an *LMP\_comb\_key* message. The random number is exchanged securely by XORing it with  $K_{init}$
4. Since both devices know  $K_{init}$  and each others' BD\_ADDR, they can compute the other party's contribution  $K_A$  ( $K_B$ )
5. The common authentication (link) key is  $K_{AB} = K_A \oplus K_B$

10.37

## Step 3: Authentication (1)

KJhole.com

1. The **verifier** device sends an *LMP\_in\_rand* message containing a random number to the **claimant**
2. The verifier and claimant devices both use the random number to initialize the  $E_1$  algorithm (modified version of SAFER+)
3. Next, the verifier sends an *LMP\_au\_rand* message containing the random number to be authenticated by the claimant
4. The claimant utilizes the  $E_1$  algorithm to encrypt this number using its authentication key  $K_{AB}$  and BD\_ADDR. It then returns the encrypted number in an *LMP\_sres* message

10.38

## Step 3: Authentication (2)

KJhole.com

5. The verifier encrypts the random number from *LMP\_au\_rand* with its authentication key and compares it with the encrypted version in *LMP\_sres*

- **Remarks:**

- Note that verifier can decide whether both sides share the same authentication key without the key ever being transmitted in clairtext
- During a mutual authentication, each device is subsequently the claimant (verifier) in two authentication procedures

10.39

## Step 4: Encryption Key Generation

- The modified SAFER+ algorithm,  $E_3$ , generates the **encryption key**,  $K_c$ , from the current authentication key
- The key size is factory preset. It may vary between 1 and 16 octets. Applications may not decrease security by reducing key size
- Currently, an encryption key size of 64 bits gives satisfying protection in most cases
- When encryption is activated by a LM command, the encryption key  $K_c$  is changed automatically

10.40

## Starting Encryption

KJhole.com

- The three following steps must be carried out before encrypted traffic can be exchanged:
  - negotiating encryption mode
  - negotiating key size
  - starting encryption

10.41

## Negotiating Encryption Mode and Key Size

- There are three different encryption modes:
  - no encryption
  - encrypt both point to point and broadcast packets
  - only encrypt point to point packets
- The Master first requests the maximum key length it can use
- If the key is within the capabilities of the Slave, the key is accepted; else the Master must try again with a shorter key

10.42

## Starting and Stopping Encryption KJhole.com

- Once the encryption mode and key size has been chosen for the link, encryption can be switched on and off
- Since encryption does not slow down traffic on a link once it has been established, there is usually no need to stop it
  - however, there may be a need to change encryption parameters and this cannot be done whilst encryption is active

10.43

## Broadcast encryption KJhole.com

- *Optional* support for encrypted broadcast traffic
- Common secret key, i.e. the Master key  $K_{\text{master}}$ , is distributed to all Slaves in a piconet
- The Master key is a temporary key used only for one session

10.44

- A device can be authenticated at link level, i.e, it can be verified that a pair of devices share a secret key derived from a PIN
- After authentication, devices can create shared keys which can be used to encrypt traffic on a link
- There are different device and security levels
- Security is most often imposed at the request of applications (Mode 2). White paper suggests that this security should be implemented through databases and a security manager