Bluetooth

Ibéria Medeiros

Departamento de Informática
Faculdade de Ciências da Universidade de Lisboa

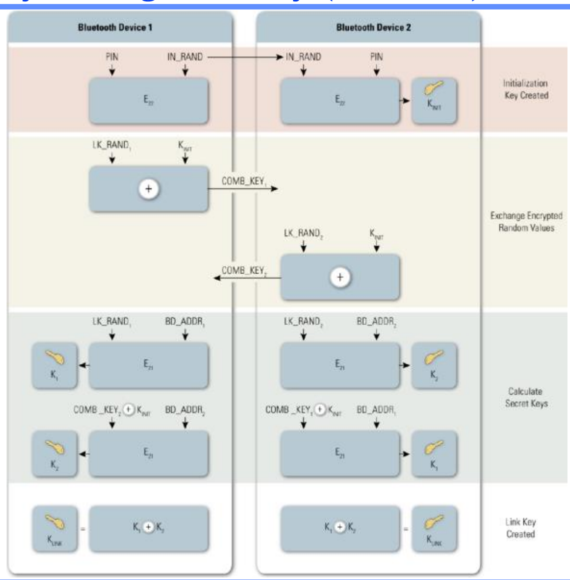
SECURITY
(BLUETOOTH <u>PRIOR</u> TO VERSION "V2.1 + EDR")

Connection Example

- *Hypothesis: HS comes from factory with a pre-set passkey that is known to the user*

1. the user places the HS is *paring mode* (e.g., by pressing a button)
2. HS indicates that is ready
3. the user makes the telephone discover other Bluetooth devices
4. the telephone broadcasts a discovery request and receives an answer from the HS
5. the HS requests authentication information from the telephone
6. the telephone finds out that it does not have an old *link key*, and starts the paring procedure
7. the telephone asks for the passkey from the user
8. a key exchange is carried out between the telephone and the HS, and a shared *link key* is derived
9. the *link key* is stored in non-volatile memory in both devices
10. HS authenticates the telephone
11. the telephone authenticates the HS
12. a message exchange is carried out and a shared *encryption key* is derived
13. from now on the communication is encrypted
14. the user removes the HS from paring mode so that new requests are not accepted

PIN/Legacy Pairing Summary (in 1 slide)



Bluetooth Implementation Aspects

- Bluetooth uses the following values to keep security at the link layer

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

unique address per device

obtained during initialization

- *BD_ADDR* is the unique address of the device
- The *authentication key* is associated with a connection and for this reason is also called a ***link key***
- The ***encryption key*** is obtained from a *link key*
- *RAND* is a random number that changes frequently and that should not be re-used during the lifetime of the *link key*

Session and Types of Keys

- A **session** corresponds to the interval of time during which the device is connected to the piconet
- A ***link key*** can be
  - ***semi-permanent*** : is stored in non-volatile memory and can be used to authenticate several sessions of the devices that share it
  - ***temporary*** : can only be used in the current session; typically is employed in point-multipoint connections with the name ***master key***, and substitutes the current *link key* temporarily
- In order to support various kinds of applications, there are 4 types of *link keys*
  - ***initialization key*** Kinit
  - ***unit key*** KA
  - ***combination key*** KAB
  - ***master key*** Kmaster
- Besides these ones there is also an ***encryption key*** Kc

Types of Keys (cont)

- The *initialization key* is used as *link key* during the initialization, while there is no *unit* or *combination keys*
- *Unit* and *combination keys* are used for the same tasks, with the only differences
  - a *unit key* is generated by only one device
  - a *combination key* is generated using information provided by both devices, and therefore is different for each pair of equipments
- The *encryption key* is different from the *link key*, therefore its size can be smaller (for example, for countries with strict rules regarding encryption)

Initialization

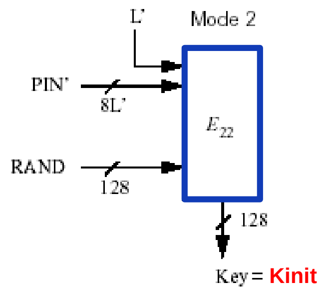
- The initialization process is done in 5 steps

  1. generation of the *initialization key* (from the secret PIN or *passkey*)
  2. generation of the *link key* (a *unit* or *combination key*)
  3. exchange of the *link key*
  4. authentication
  5. generation of the *encryption key*

- After the generation of a *link key*, the key is stored locally so that steps 1 - 3 no longer have to be performed
- A new *encryption key* is however created for each connection
- If the *link key* is lost then the initialization process has to be carried out from the beginning

Generation of the *Initialization Key, Kinit*

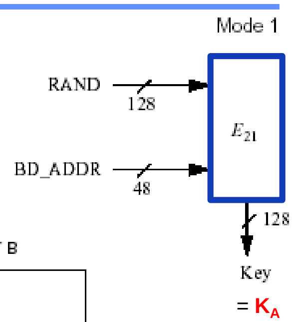
- The initialization key is generated from:
  - the BD_ADDR that wants to connect
  - a PIN (or *passkey*) with 1-16 bytes
  - the size of PIN in bytes
  - a random RAND created by the device that is going to verify

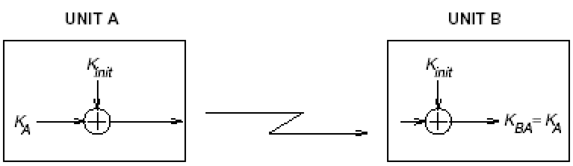


- E22 is an algorithm from Bluetooth
- PIN´ has a maximum of 16 bytes based on the PIN and BD_ADDR (if the PIN has 16 bytes then PIN´ = PIN; otherwise, the bytes of BD_ADDR are used, such PIN´ = BD_ADDR || PIN and size of PIN´ ≤ 16)
- L´ = min(16, L+6), where L is the number of bytes of PIN
- The initialization key *Kinit* is discarded after the link key exchange (step 3)

Generation of a *Unit Key, KA*

- The unit key is generated from
  - a random RAND
  - the address of the device BD_ADDR
  - uses algorithm E21
- The key is created by one of the devices, and then is distributed to the other one



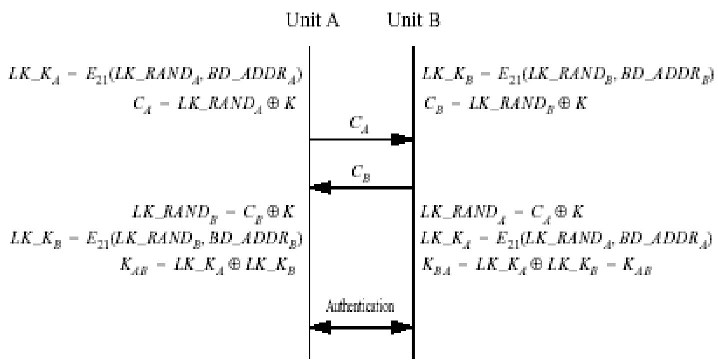


- Typically, the original device will only keep this key and use it for all connections with other equipments
- The device that received the key will also keep it, but only to protect communications with the original device

Security Considerations with Unit Keys

- Unit keys should be avoided whenever possible, but for certain cases it might be the only option (for example, small memories or devices that need to interact with many equipments)
- A device with a unit key will re-use it for communications with all other devices
- Two attacks can be performed
  - an equipment that obtains a unit key can listen to every communication of the corresponding device
  - the equipment that got the key can personify the device to other peers

Generation of a *Combination Key, KAB*

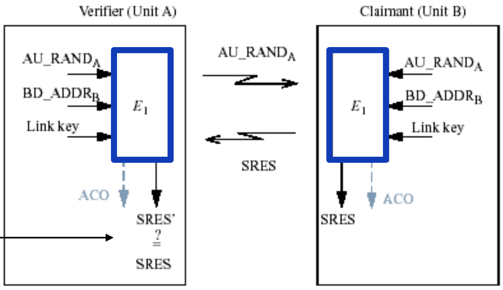


1. Two pseudo-random sub-keys LK_KA e LK_KB are calculated like the unit keys
2. Random LK_RANDA and LK_RANDB are exchanged securely by XORing them with *Kinit*
3. The combination key is the XOR of the two contributions

Security Considerations with Combination Keys

- Combination keys offer a higher level of security than unit keys because they are calculated using random values from two devices, which means distinct peers will have different keys

Authentication



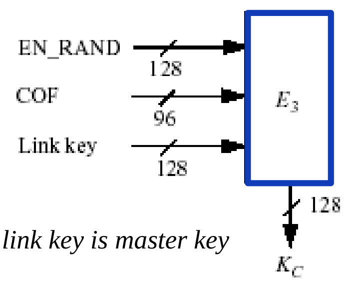
If they are equal authentication went well

- Uses a challenge-response scheme, which in case of mutual authentication is carried out in both directions
- The link key being used is either a unit or a combination key
- As result of a correct authentication, the parameter *Authentication Ciphering Offset (ACO)* is returned so that it can used to generate the encryption key
- In case of auth failure, it can only be retried after a period that doubles for each try

Security Considerations

- All security is based on the passkeys, and the following attack can be performed
  - the adversary listens to all message exchanges during the creation of the initialization key, combination keys and authentication
  - the adversary tries various passkeys until she can generate the values observed during the authentication (when this happens, with high probability she has guessed the passkey)
- Therefore, if weak passkeys are being utilized, then it is necessary to perform the initialization process **in a private place**, where the adversary cannot listen to the initial exchanges
- After the initialization is done, future authentications are based on the stored link key and therefore the **security problem no longer exists** (the key is 128 bits)

Generation of the *Encryption Key, Kc*



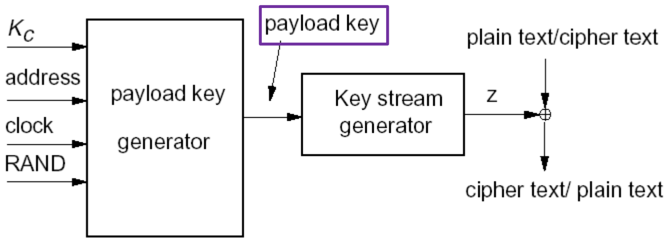
- The encryption key is created from
  - a random EN_RAND
  - the current link key
  - the *Ciphering Offset number (COF)* with 96 bits

$$COF = \begin{cases} BD\_ADDR \cup BD\_ADDR, & \text{if the link key is master key} \\ ACO, & \text{otherwise} \end{cases}$$

- Uses algorithm E3 that basically calculates an hash
- EN_RAND has to be exchanged prior to this operation
- If the link key is a master key, the master address *BD_ADDR* is used to get the COF

Data Encryption

- Encryption protects the frame data part (and not the header)
- A stream cipher algorithm is employed E0
- E0 is re-synchronized for each transmitted frame, and is composed by three parts: a per-frame *payload key*; a pseudo-random generator of bits; and the XOR operation with the data



Bluetooth Security Issues

- *Bluejacking*: sending of unsolicited messages to Bluetooth-enabled devices (sending a vCard which typically contains a message in the name field to another Bluetooth device)
- *Bluesnarfing*: gain access to data stored in the phone without alerting the user (by exploring a firmware flaw in older devices)
- *Bluebugging*: allows an attacker to access the mobile phone commands using Bluetooth without alerting the phone´s user (e.g., to initiate phone calls, read and write phone´s contacts, listen to phone calls, connect to the Internet)
- *DoS attacks*: waste the victim´s power by sending unsolicited messages
- *Backdoor attack*: an attack that exploits a previous trust relationship between two devices (the link key has to be removed to prevent the attack)
- *Other protocol attacks*: try to recover the PIN as explained; or break the E0 encryption algorithm

Bibliography

- *Guide to Bluetooth Security*, NIST Special Publication 800-121, Revision 1, June 2012
- S. Aissi et al, *Security for Mobile Networks and Platforms*, Artech House, 2006 (section 3.3; chapter 11)