Bluetooth 5.4

Study of how pairing algorithm work and what is new in version 5.4 and how encryption is done

Tirelli Andrea - 191979

Bluetooth - Intro

- Pairing
- BT 5.4 - PAwR Encryption

Pairing - Intro

The first step in establishing a Bluetooth connection is the pairing process.

"Two devices discovering each other and establishing a connection."

- Discovery
- **Pairing request & response**
- **Authentication**

Pairing - Additional knowledge (non crypto)

Network form by two or more devices is called **piconet**.

➔ network form by multiple piconet is called **scatternet**.

Type of links:

- Synchronous Connection-Oriented (SCO)
- **Asynchronous Connection-Less** (ACL)

Pairing - Request/Response packet

Field	Code (1 Byte)	IO Cap (1 Byte)	OOB DF (1 Byte)	AuthReq (1 Byte)					Maximum Encryption Key Size (1 Byte)	Initiator Key Distribution (1 Byte)	Responder Key Distribution (1 Byte)
Sub-define				BF	MITM	SC	KP	Reserved			
Bits*	8	8	8	2	1	1	1	3	8	8	8

Table 1 Pairing Request/Response

*Bit order is LSB to MSB.

- IO Cap
- OOB DF
- MitM
- SC
- Max Encryption Key Size

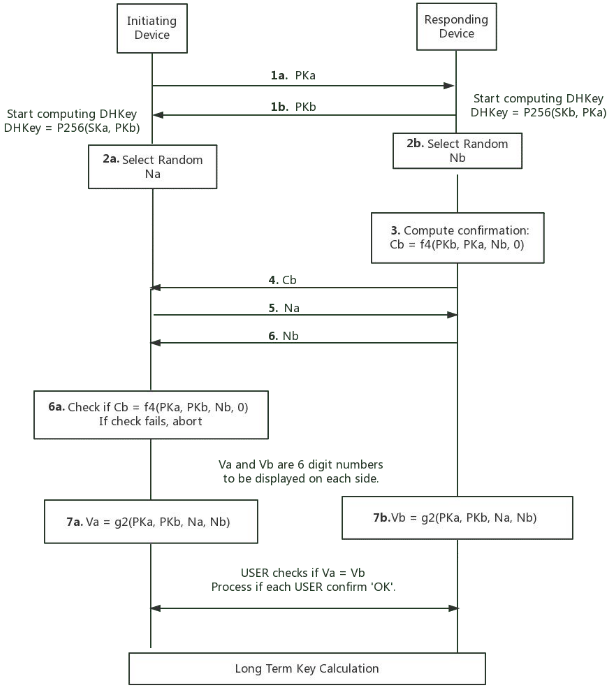
Pairing - Authentication algorithm

	Initiator				
Responder	**DisplayOnly**	**Display YesNo**	**Keyboard Only**	**NoInput NoOutput**	**Keyboard Display**
Display Only	Just Works Unauthenti-cated	Just Works Unauthenti-cated	Passkey Entry: responder displays, ini-tiator inputs Authenti-cated	Just Works Unauthenti-cated	Passkey Entry: responder displays, ini-tiator inputs Authenti-cated
Display YesNo	Just Works Unauthenti-cated	Just Works (For LE Legacy Pairing) Unauthenti-cated Numeric Comparison (For LE Secure Con-nections) Authenti-cated	Passkey Entry: responder displays, ini-tiator inputs Authenti-cated	Just Works Unauthenti-cated	Passkey Entry (For LE Legacy Pairing): responder displays, ini-tiator inputs Authenti-cated Numeric Comparison (For LE Secure Con-nections) Authenti-cated
Keyboard Only	Passkey Entry: initia-tor displays, responder inputs Authenti-cated	Passkey Entry: initia-tor displays, responder inputs Authenti-cated	Passkey Entry: initia-tor and responder inputs Authenti-cated	Just Works Unauthenti-cated	Passkey Entry: initia-tor displays, responder inputs Authenti-cated
NoInput NoOutput	Just Works Unauthenti-cated	Just Works Unauthenti-cated	Just Works Unauthenti-cated	Just Works Unauthenti-cated	Just Works Unauthenti-cated
Keyboard Display	Passkey Entry: initia-tor displays, responder inputs Authenti-cated	Passkey Entry (For LE Legacy Pairing): initiator dis-plays, responder inputs Authenti-cated Numeric Comparison (For LE Secure Con-nections) Authenti-cated	Passkey Entry: responder displays, ini-tiator inputs Authenti-cated	Just Works Unauthenti-cated	Passkey Entry (For LE Legacy Pairing): initiator dis-plays, responder inputs Authenti-cated Numeric Comparison (For LE Secure Con-nections) Authenti-cated

- Legacy Pairing: Just Work
- Secure Pairing: Numeric Comparison

NOTE: Legacy Pairing != Unauthenticated

Pairing - Just Work & Numeric Comparison (1)



Note:
* P256(), Elliptic-Curve Diffie-Hellman fuction.
* f4() is used to generate confirm values during the pairing process.
* g2() is used to generate the 6-digit numeric comparison values during authentication process.
* For details about cryptographic toolbox, please refer to Bluetooth Core Spec v5.0, Vol 3, Part H, Section 2.2.

Just Work and Numeric Comparison:

1. DHKey exchange
2. Generation of nonce
3. Calculate commit
4. Share commit(s)
5. Check commit(s)

ONLY Numeric Comparison:

+ Verification (and so authentication) of the commit value

Pairing - Just Work & Numeric Comparison (2)

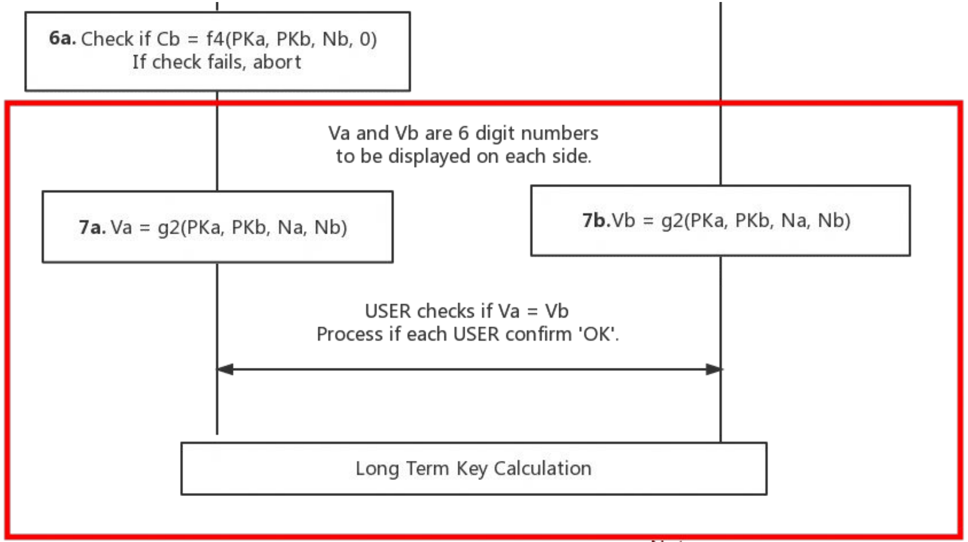
Just Work, the problem:

"When Just Works is used, the commitment checks (steps 7a and 7b) are not performed, and the user is not shown the 6-digit values."

➔ Possible attack: **Man-in-the-Middle**

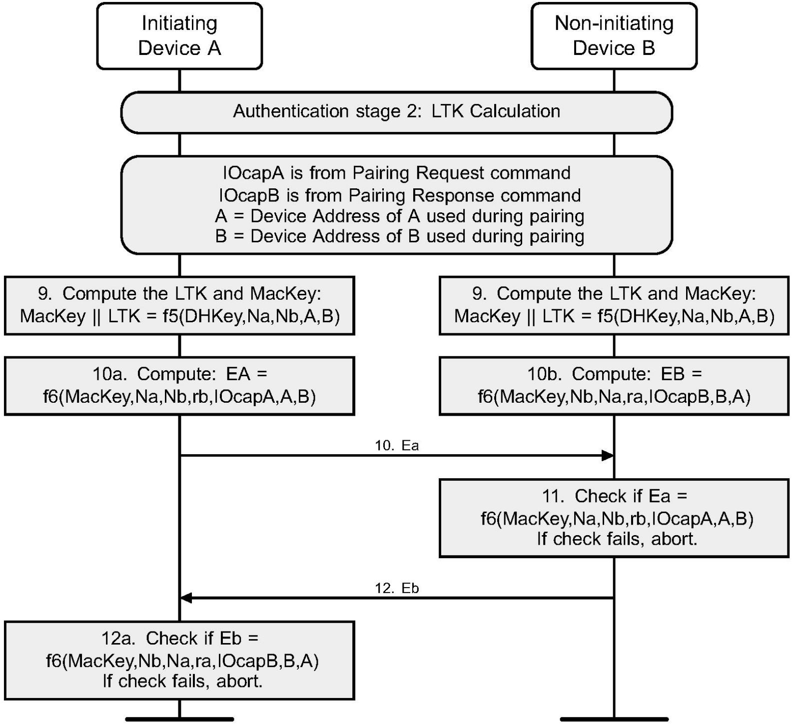
https://www.bluetooth.com/wp-content/uploads/Files/Specification/HTML/Core-54/out/en/host/security-manager-specification.html

Pairing - Just Work & Numeric Comparison (3)



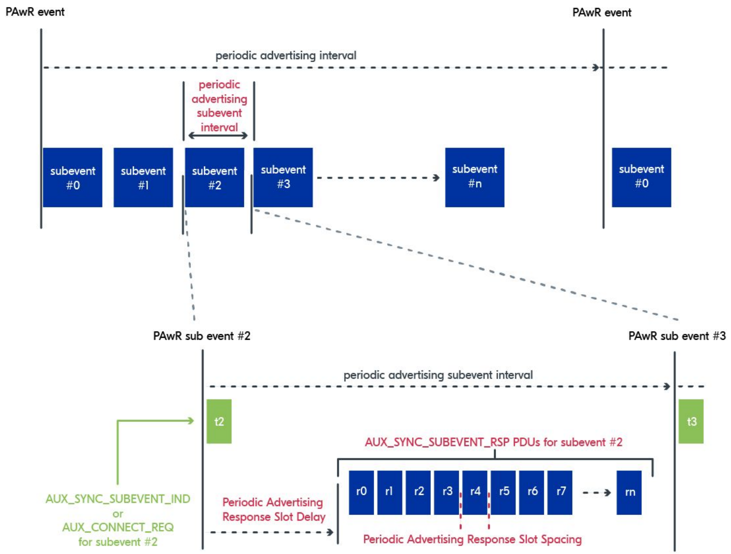
Note:
* P256(), Elliptic-Curve Diffie-Hellman fuction.
* f4() is used to generate confirm values during the pairing process.
* g2() is used to generate the 6-digit numeric comparison values during authentication process.
* For details about cryptographic toolbox, please refer to Bluetooth Core Spec v5.0, Vol 3, Part H, Section 2.2.

Pairing - Long Term Key (LTK)



It is important to highlight that after the **authentication** phase terminate with success, it is **not possible** for an attacker to attack the scheme.

PAwR - Intro



Main goal -> Define structures and algorithm that is required for this type of communication.

- (BT 5.4) support for response by the slaves of the network.

Before 5.4 (so in BT ≤ 5.3) every manufacturer could implements it's own way to do that.

PAwR - Packet structure

Encrypted Data AD type

- Encrypted Data:
  - Randomizer, used to formulate the *nonce*
  - 2 AD
    - ESL*
    - Local Name*
  - MIC, Message Integrity Check, Bluetooth name definition for the MAC
- Unencrypted Data:
  - Flags

**This AD might be different, this is the case for and ESL environment*

PAwR - Security Guarantees

- **Confidentiality**
- **Authentication**

This indicates to apply an AEAD scheme…

But it is not possible to apply the notions given to us by literature:

- **Encrypt-then-MAC**
- **2 Keys**
  - 1 key for authentication - AES-CMAC
  - 1 key for confidentiality - AES-CBC

PAwR - CCM mode (1)

Scheme follow the mode: **Authenticate-then-Encrypt**

CBC-MAC: authentication

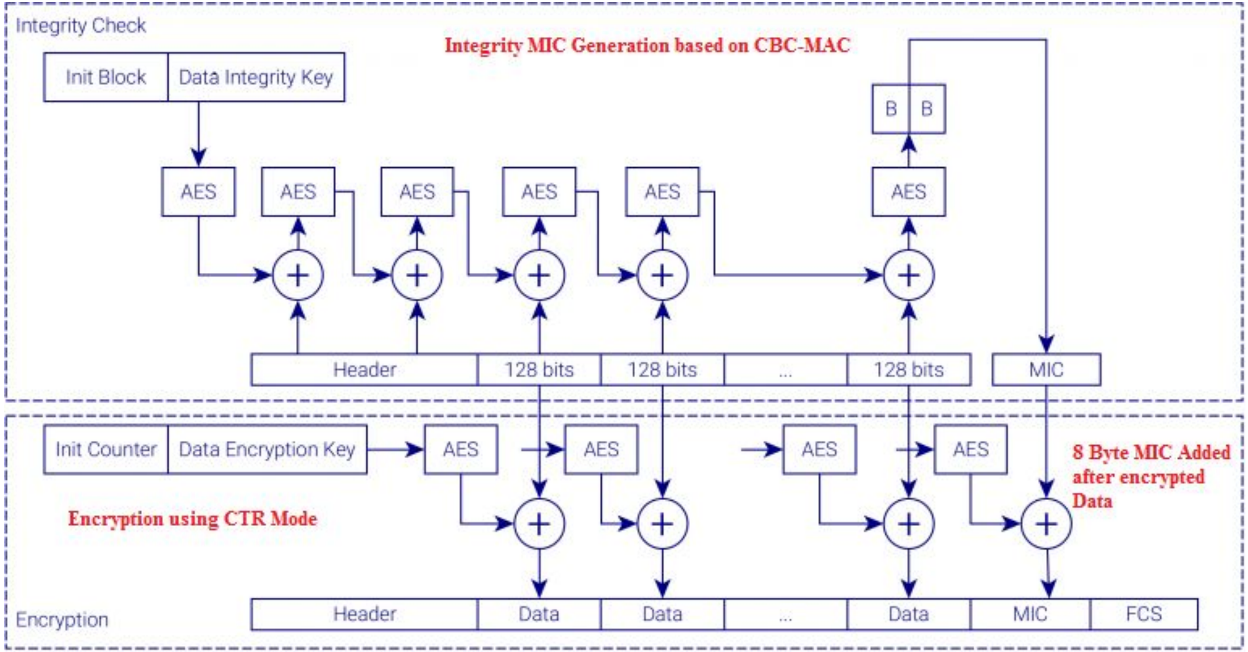
CTR Mode: encryption

Important notes:

"*The nonce of CCM must be carefully chosen to never be used more than once for a given key. This is because CCM is a derivation of counter (CTR) mode and the latter is effectively a stream cipher.*"

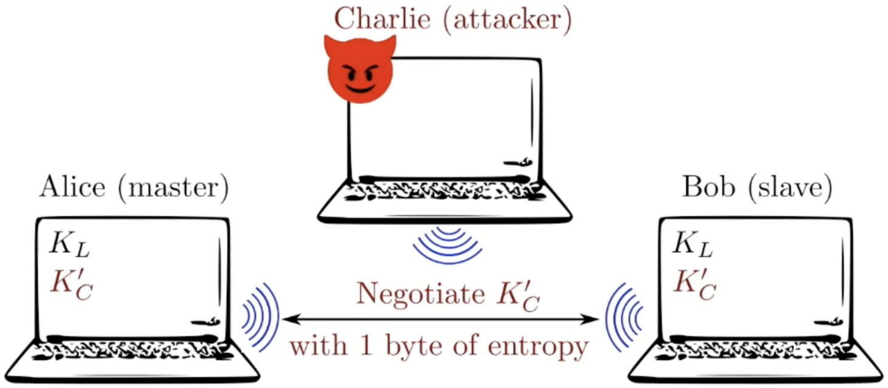
Housley, Russ (December 2005). "rfc4309". IETF: 3. AES CCM employs counter mode for encryption. As with any stream cipher, reuse of the same IV value with the same key is catastrophic.

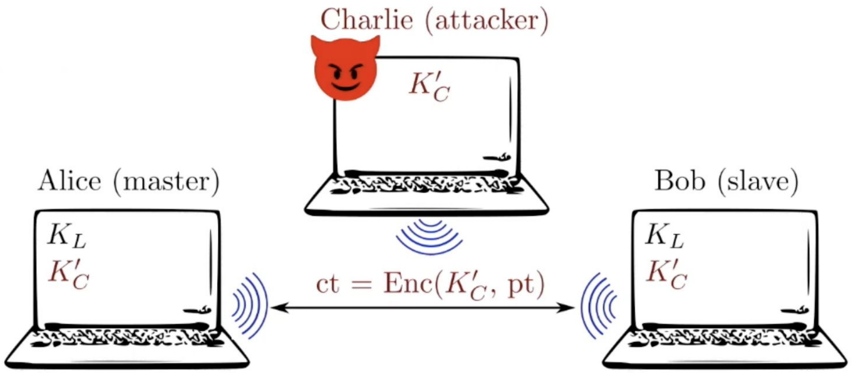
PAwR - CCM mode (2)



Notice how CTR Mode use **K** and **counter**, this form a sort of **Stream cipher** to encrypt data.

KNOB - Key Negotiation of Bluetooth Attack (1)





Idea: negotiate a lower **size key** (1byte)

1 byte key ciphertext **ct=Enc(K’c, pt)**

Entropy negotiation is **not encrypted** and **not authenticated**.

KNOB - Key Negotiation of Bluetooth Attack (2)

"For the encryption algorithm, ***the key size (N) 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****."*

https://www.youtube.com/watch?v=v9Xg9XcnNh0