BLE 應用安全與分析
BLE Application Security and Analysis

查士朝
台灣科技大學資訊管理系

1 從物聯網安全開始談起

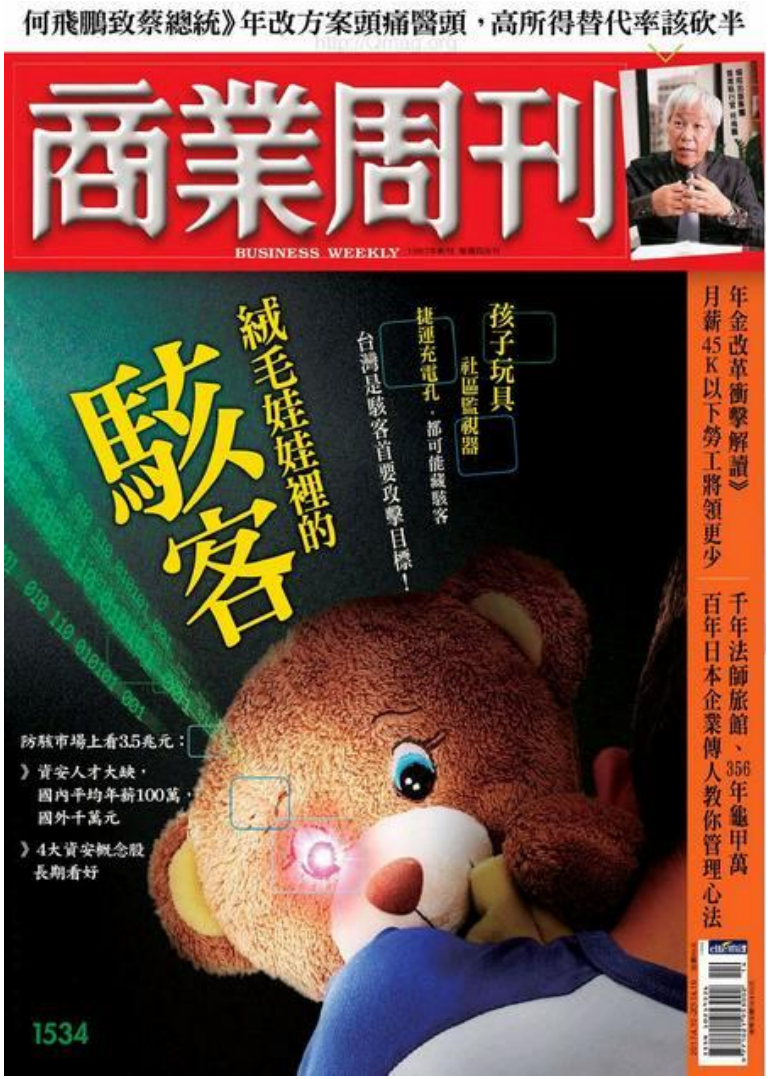
2 BLE 廣播與弱點

3 BLE 連線與弱點

4 結論與未來方向

1 從物聯網安全開始談起

何飛鵬致蔡總統》年改方案頭痛醫頭，高所得替代率該砍半



The Switch

Hello (hackable) Barbie

By Andrea Peterson December 4, 2015

Hello Barbie on display at the Mattel showroom at the North American International Toy Fair in New York. (Mark Lennihan/AP)

Toys that talk back are some of the hottest holidays gifts this year. And they may soon be hot items for hackers.

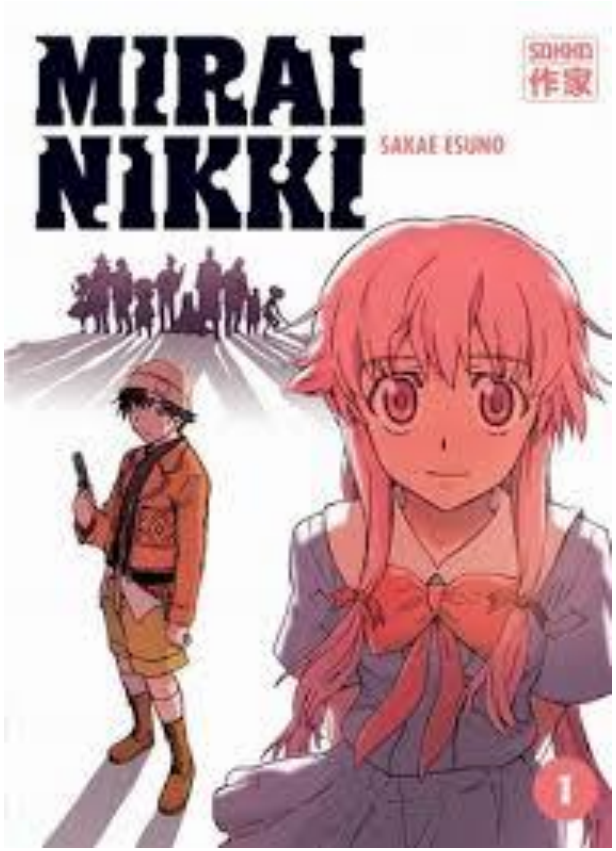
Cybersecurity researchers uncovered a number of major security flaws in systems behind Hello Barbie, an Internet-connected doll that listens to children and uses artificial intelligence to respond. Vulnerabilities in the mobile app and cloud storage used by the doll could have allowed hackers to eavesdrop on even the most intimate of those play sessions, according to a report released Friday by Bluebox Security and independent security researcher Andrew Hay.

https://www.washingtonpost.com/news/the-switch/wp/2015/12/04/ hello-hackable-barbie/?utm_ term=.986338ab8576

https://www.rsaconference.com/writable/presentations/file_upload/sbx1-r08-barbie-vs-the-atm-lock.pdf

推出物聯網裝置時，有時帶來的安全與隱私威脅會超乎想像

Mirai – 針對 IoT 的病毒



Article Talk Read Edit View history Search Wikipedia

Mirai (malware)

From Wikipedia, the free encyclopedia

Mirai (Japanese for "the future", 未来) is malware that turns computer systems running Linux into remotely controlled "bots", that can be used as part of a botnet in large-scale network attacks. It primarily targets online consumer devices such as remote cameras and home routers.[1] The Mirai botnet was firstly found in August 2016[2][3] by MalwareMustDie,[4] a whitehat malware research group, and has been used in some of the largest and most disruptive distributed denial of service (DDoS) attacks, including an attack on 20 September 2016 [5] on computer security journalist Brian Krebs's web site, an attack on French web host OVH,[6] and the October 2016 Dyn cyberattack.[7][8][9] According to a leaked chat log between Anna-senpai and Robert Coelho, Mirai was named after the 2011 TV anime Mirai Nikki.[10]

Mirai

Original author(s)	"Anna-senpai" (online pseudonym)
Repository	github.com/James-Gallagher/Mirai
Written in	C (agent), Go (controller)
Operating system	Linux
Type	Botnet

The source code for Mirai has been published in hacker forums as open-source.[11] Since the source code was published, the techniques have been adapted in other malware projects.[12]

Contents [hide]

1 Malware

在 IoT 的領域，使用者的疏忽常會成為可趁之機

https://en.wikipedia.org/wiki/Mirai_(malware)

A hacker has released computer source code that allows relatively unsophisticated people to wage the kinds of extraordinarily large assaults that recently knocked security news site KrebsOnSecurity offline and set new records for so-called distributed denial-of-service attacks.

FURTHER READING

Why the silencing of KrebsOnSecurity opens a troubling chapter for the 'Net

KrebsOnSecurity's Brian Krebs reported on Saturday that the source code for "Mirai," a network of Internet-connected cameras and other "Internet of things" devices, was published on Friday. Dale Drew, the chief security officer at Internet backbone provider Level 3 Communications, told Ars that Mirai is one of two competing IoT botnet families that have recently menaced the Internet with record-breaking distributed denial-of-service (DDoS) attacks—including the one that targeted Krebs with 620 gigabits per second of network traffic, and another that hit French webhost OVH and reportedly peaked at more than 1 terabit per second

Until now, the botnets created with the newer and technically more sophisticated Mirai have been greatly outnumbered by those based on its rival Bashlight, with about 233,000 infected devices versus 963,000 respectively. Friday's release could allow the smaller and more disciplined Mirai, which Russian antivirus provider Dr. Web briefly profiled last week, to go mainstream. That, in turn, could turn the mass compromise of cameras and other Internet-connected devices into a full-blown epidemic that could push record DDoSes to ever-higher volumes. In an e-mail to Ars, Drew wrote:

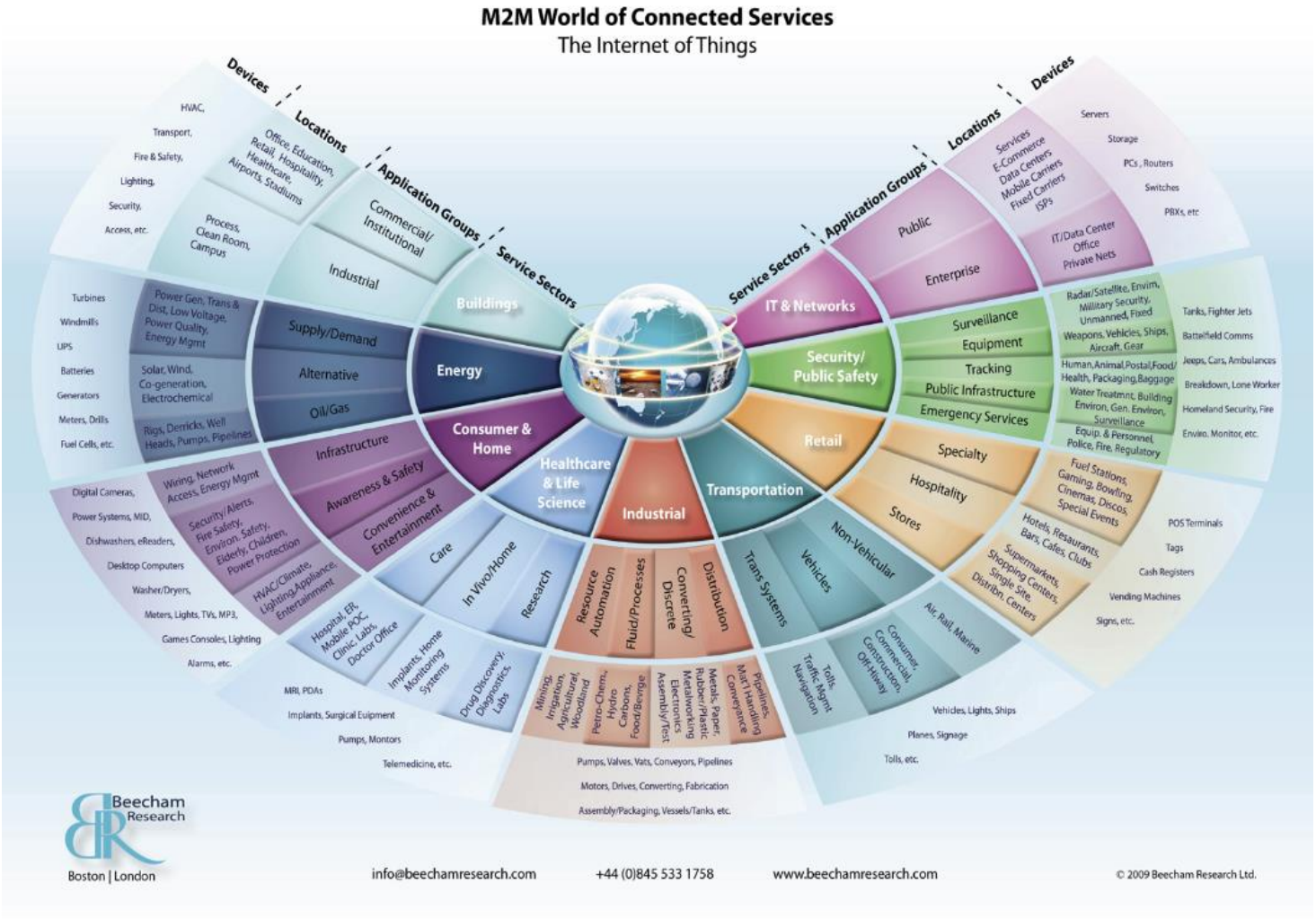
https://arstechnica.com/security/2016/10/brace-yourselves-source-code-powering-potent-iot-ddoses-just-went-public/

are described in today's announcement. These new enhancements allow attacks detected by DDoS Secure at the network and application-layer to be stopped closer to the source by using networking protocols to make the Juniper MX Series routers function as enforcement points.

This approach provides enterprises and service providers a more efficient way of stopping the volumetric attacks that can potentially cripple a network. It also mitigates other popular DDoS attack methods, including inside-out Domain Name System (DNS) reflection and amplification attacks, as well as the negative effects that botnet-infected devices can have on the user experience for a service provider's customers. According to Infonetics Research, new varieties of amplification attacks are pushing the boundaries of mitigation performance and driving increased investment in DDoS prevention. Examples include the 2013 DNS amplification attack aimed at Spamhaus that topped 300G, and the NTP amplification attack earlier this year that exceeded 400G.(1)

http://newsroom.juniper.net/press-releases/juniper-networks-delivers-the-first-holistic-distributed-denial-of-service-ddos-nyse-jnpr-1131859

IoT 安全與隱私問題所以複雜，因為涵蓋各階段與領域



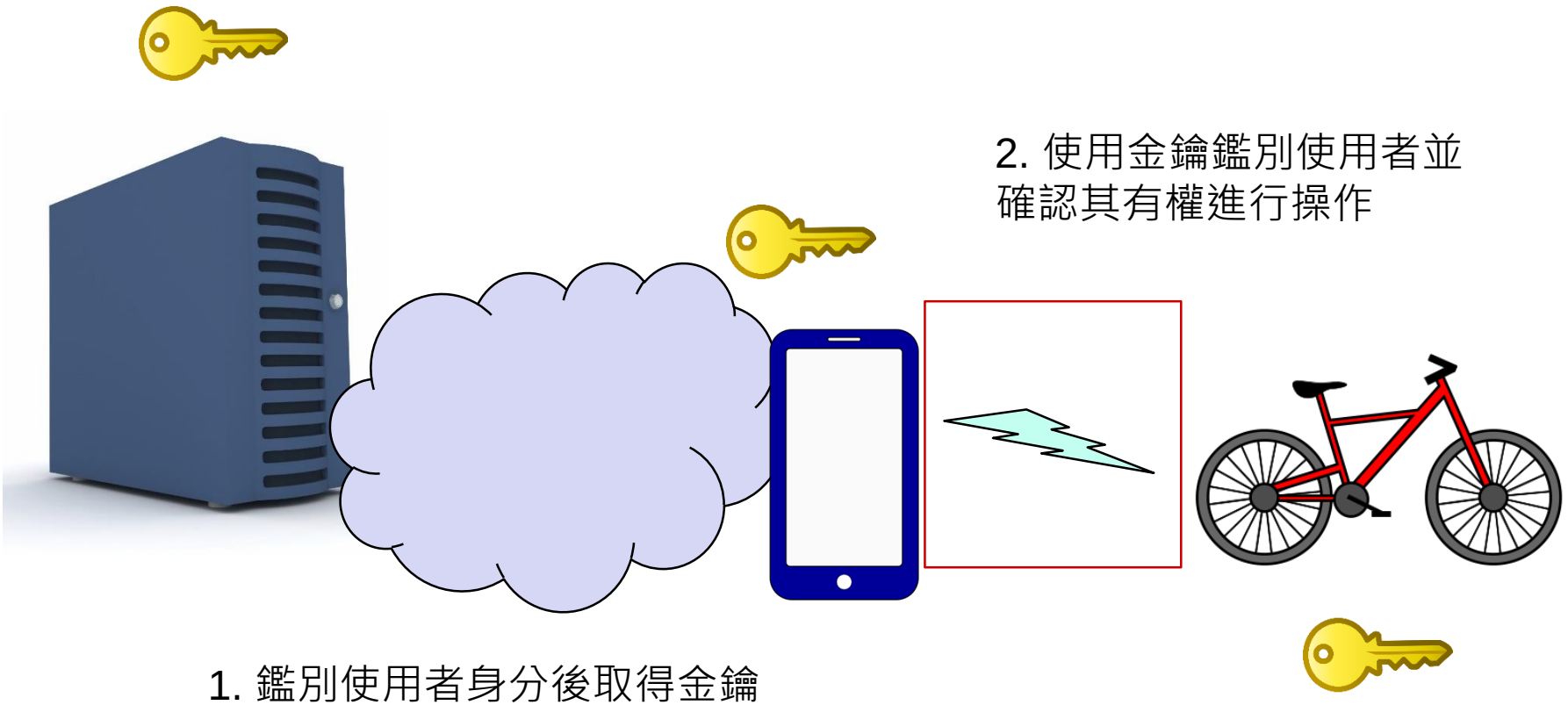
IoT 安全和隱私議題範圍太廣，在學習時需要鎖定幾個應用

- 例如歐盟 Article 29 資料保護工作小組在「對IoT 發展的意見書 Opinion 8/2014 on the on Recent Developments on the Internet of Things」當中，就鎖定以下三個使用情境來檢視安全與隱私議題
  - 穿戴式運算 (Wearable Computing)
  - 量化自身 (Quantified Self)
  - 家庭自動化 (Home automation)

要了解一個很複雜的問題時，有時需要把問題做分割，並針對特定問題去解決

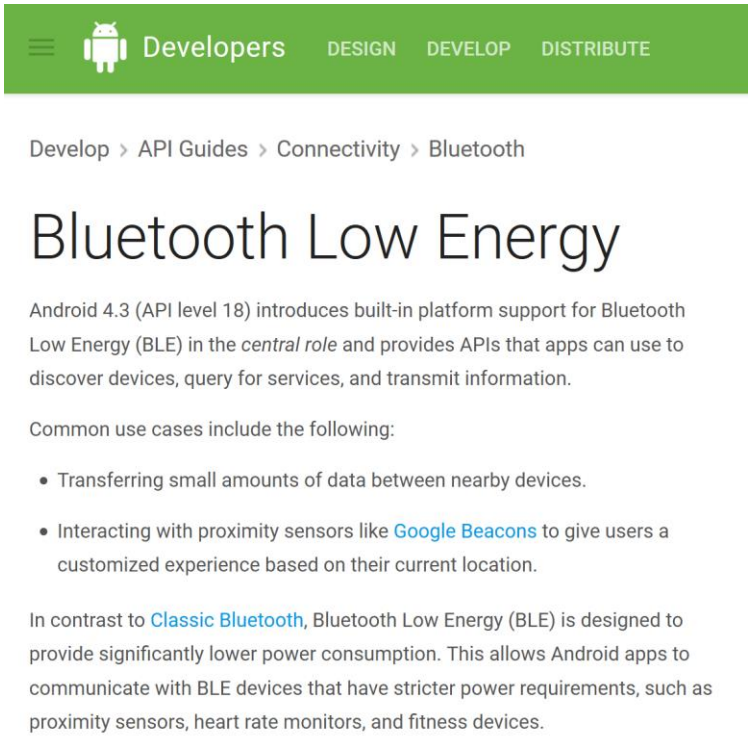
2 BLE 廣播與弱點

就算是一個特定應用，仍會有很多不同面向的議題



2. 使用金鑰鑑別使用者並確認其有權進行操作

1. 鑑別使用者身分後取得金鑰

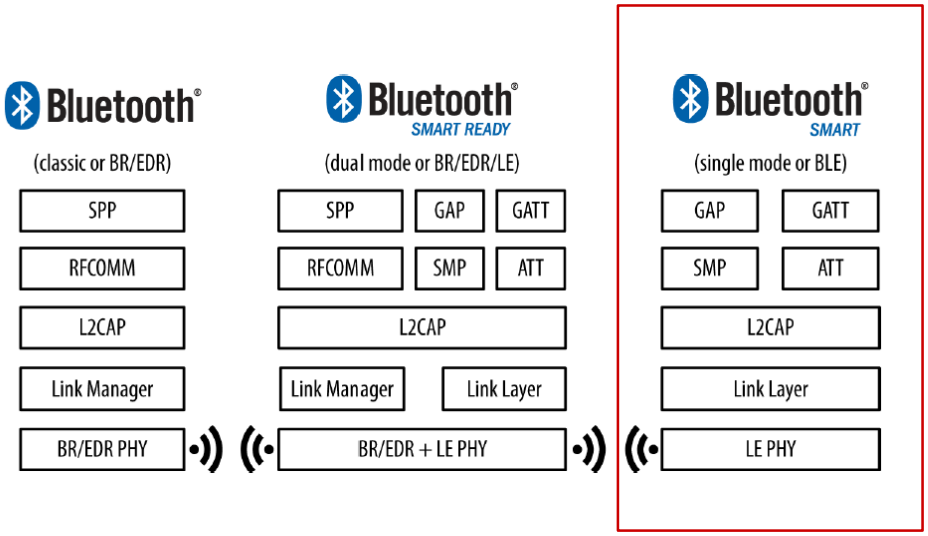


https://developer.android.com/guide/topics/connectivity/bluetooth-le.html

https://developer.apple.com/bluetooth/

BLE (Bluetooth Low Energy)

- 在 2010 年，伴隨藍牙 4.0 標準推出
- BLE 的設計目的，就是希望能夠透過一顆鈕扣電池就可以在裝置的生命週期中都不用換電池
- 依支援標準不同，可以分為三種：
  - 僅支援傳統 Bluetooth
  - 僅支援 BLE
  - 同時支援傳統 Bluetooth 與 BLE



Source: Getting Started with Bluetooth Low Energy - Tools and Techniques for Low-Power Networking

BLE 裝置的類型

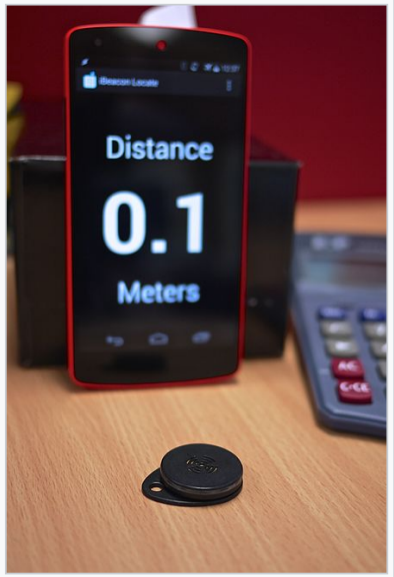
- 廣播 (Broadcaster) 與接收者 (Receiver)
- 中央 (Central) 與周邊 (Peripheral)

iBeacon 廣播與接收

iBeacon

From Wikipedia, the free encyclopedia

iBeacon is a protocol developed by Apple and introduced at the Apple Worldwide Developers Conference in 2013.[1] Various vendors have since made iBeacon-compatible hardware transmitters – typically called beacons – a class of Bluetooth low energy (BLE) devices that broadcast their identifier to nearby portable electronic devices. The technology enables smartphones, tablets and other devices to perform actions when in close proximity to an iBeacon.[2][3]



Smartphone detecting an iBeacon transmitter

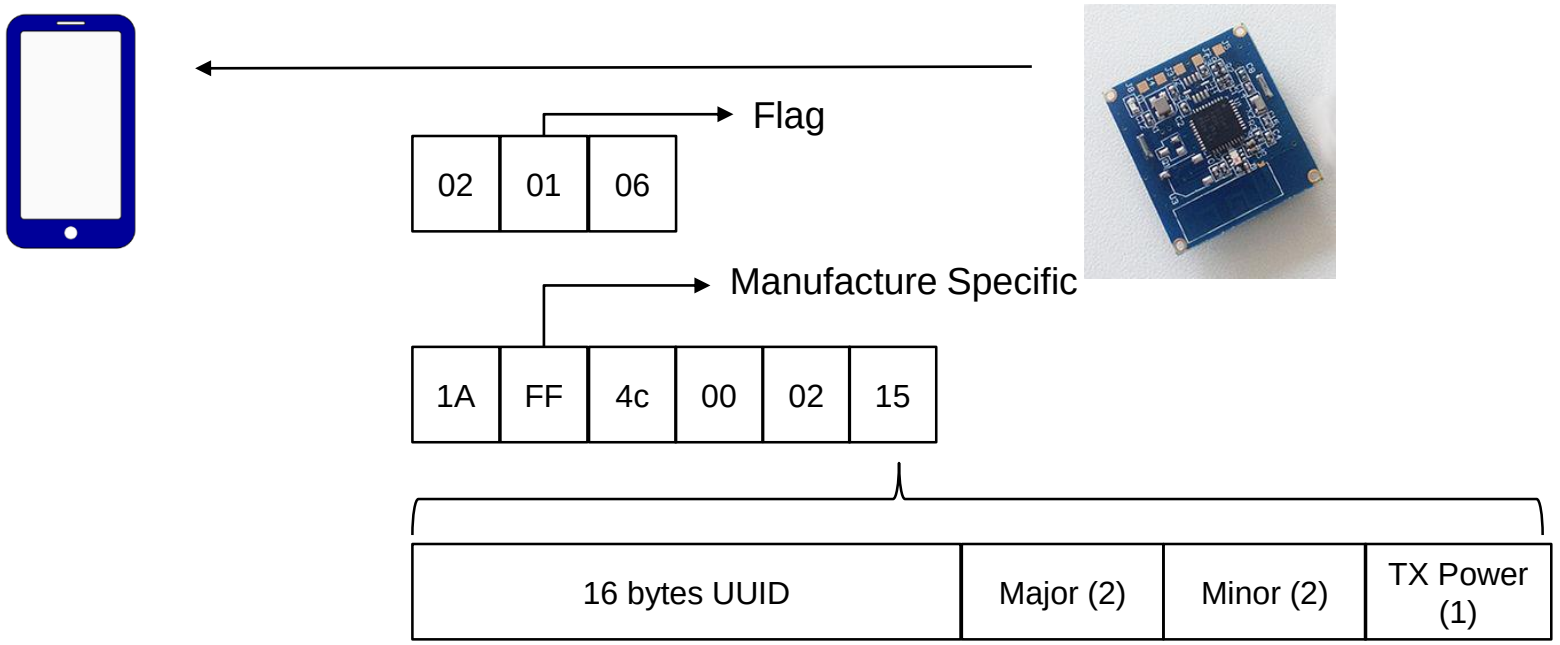
iBeacon is based on Bluetooth low energy proximity sensing by transmitting a universally unique identifier[4] picked up by a compatible app or operating system. The identifier and several bytes sent with it can be used to determine the device's physical location,[5] track customers, or trigger a location-based action on the device such as a check-in on social media or a push notification.

iBeacon can also be used with an application as an indoor positioning system,[6][7][8] which helps smartphones determine their approximate location or context. With the help of an iBeacon, a smartphone's software can approximately find its relative location to an iBeacon in a store. Brick and mortar retail stores use the beacons for mobile commerce, offering customers special deals through mobile marketing,[9] and can enable mobile payments through point of sale systems.

Another application is distributing messages at a specific Point of Interest, for example a store, a bus stop, a room or a more specific location like a piece of furniture or a vending machine. This is similar to previously used geopush technology based on GPS, but with a much reduced impact on battery life and better precision.

https://en.wikipedia.org/wiki/IBeacon

iBeacon 廣播與接收



廣播有可能偽造嗎？
如何驗證正確性？

Flag
01: LE Limited Discoverable Mode 02: LE General Discoverable Mode 04: BR/EDR not supported 08: Simultaneous LE and BR/EDR (Controller) 10: Simultaneous LE and BR/EDR (Host)

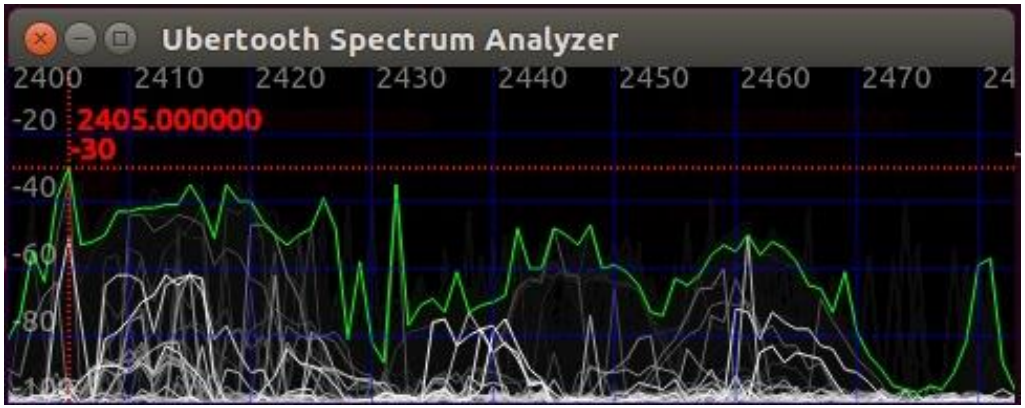
Company ID
0001: Nokia 0002: Intel 0003: IBM 0006: Microsoft 004C: Apple, Inc. 00E0: Google

使用工具側錄 BLE 封包 – 以 Ubertooth One 為例

https://github.com/greatscottgadgets/ubertooth/wiki/Build-Guide

安裝程序摘要

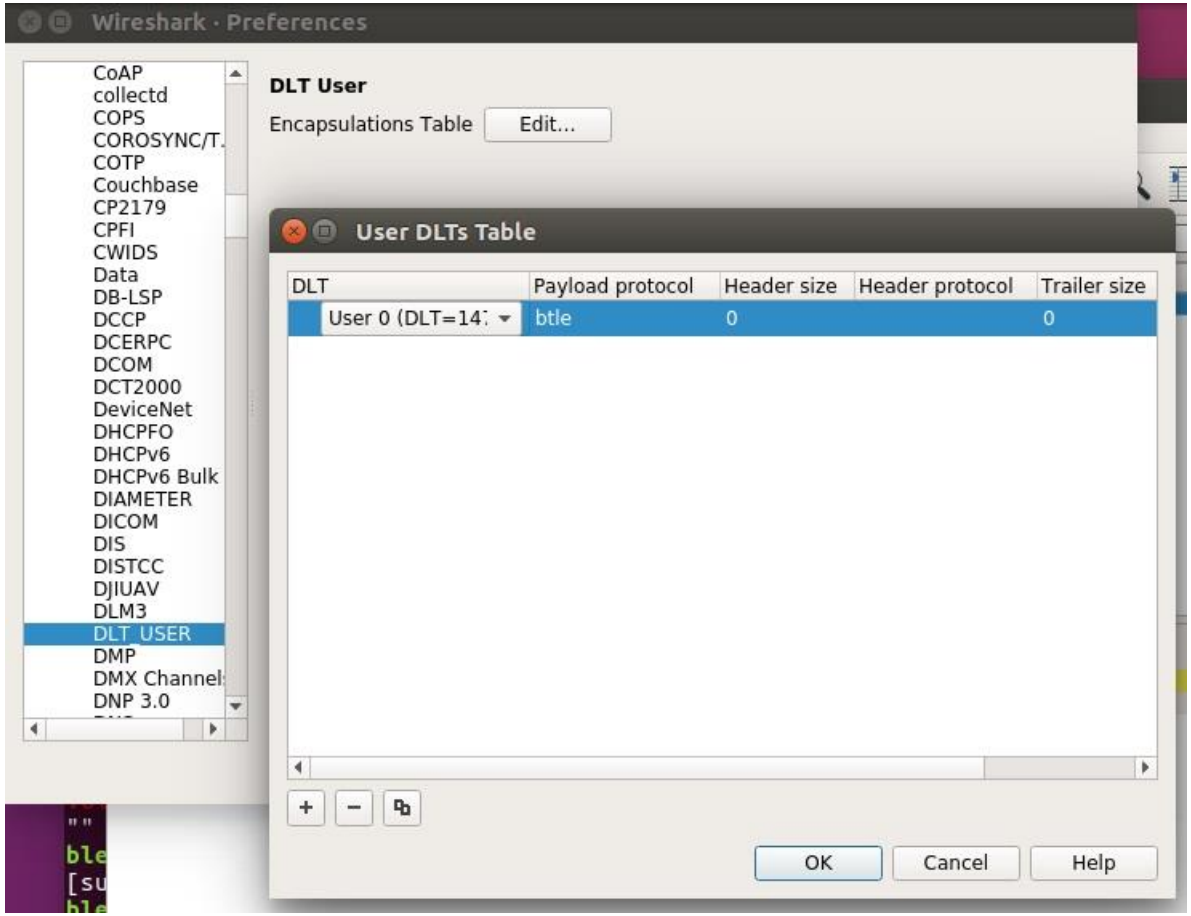
- 先裝一些必要的套件
- 下載、編譯與安裝 libbtbb
- 下載、編譯與安裝 Ubertooth tools
- 安裝 wireshark
- 編譯與安裝 BT LE 與 BT BR/EDR
- 必要時更新韌體
- 執行 ubertooth-spec-ui 來看是否有成功安裝



在 Wireshark 上直接檢視封包

- 可以建立 pipe，然後透過 Capture -> Options 命令叫起視窗後，按下 Manage Interfaces，然後新增所建立的 pipe
  - mkfifo /tmp/pipe
- 透過 ubertooth-btle -f -c /tmp/pipe 指令將資料吐到 pipe
- 如果沒有處理 DLT=147 的型態，需要
  - 到 Edit -> Preferences叫起視窗
  - 選單中選擇 Protocols -> DLT_USER
  - 按下 Edit (Encapsulations Table) 按鈕
  - 在 Payload Protocol 中輸入 btle

https://github.com/greatscottgadgets/ubertooth/wiki/Capturing-BLE-in-Wireshark


Wireshark · Preferences
CoAP
collectd
COPS
COROSYNC/T.
COTP
Couchbase
CP2179
CPFI
CWIDS
Data
DB-LSP
DCCP
DCERPC
DCOM
DCT2000
DeviceNet
DHCPFO
DHCPv6
DHCPv6 Bulk
DIAMETER
DICOM
DIS
DISTCC
DJIUAV
DLM3
DLT_USER
DMP
DMX Channel
DNP 3.0
DLT User
Encapsulations Table
Edit...
User DLTs Table
DLT
Payload protocol
Header size
Header protocol
Trailer size
User 0 (DLT=147)
btle
0
0
OK
Cancel
Help

ubertooth-btle 主要參數

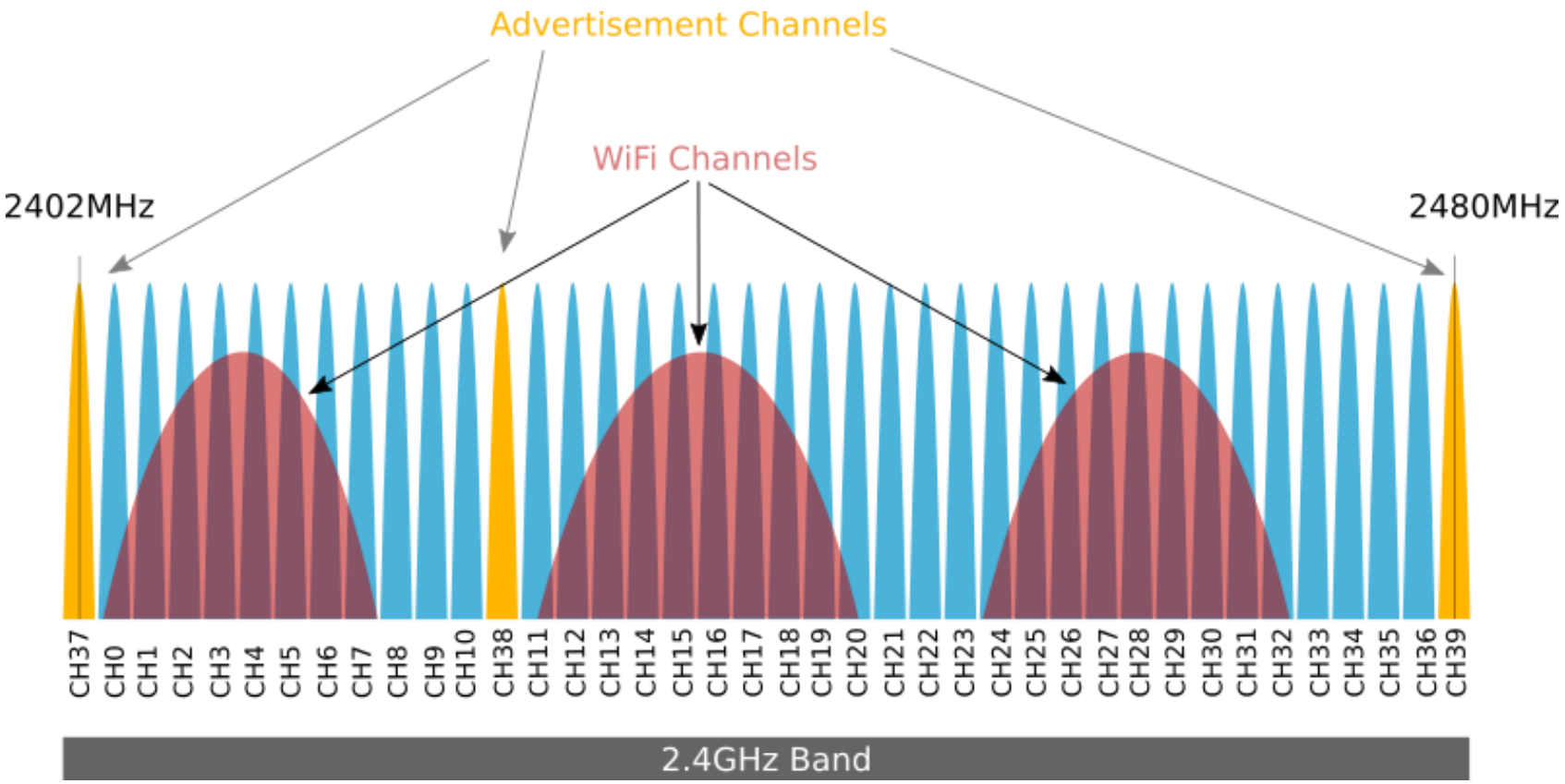
- -h：列出指令說明
- -f：發現連線時就會一直去追封包
- -p：會儘可能去抓所有封包
- -a<address>：針對特定 MAC 位置 (example: -a8e89bed6)
- -t<address>：針對特定 MAC 目標 (example: -t22:44:66:88:aa:cc)

- -c<filename>：將封包抓為 PCAP 格式的檔案 (DLT_PPI)

跳頻的概念

- 在 Bluetooth 5 以前，有 37 個資料傳輸頻道 3 個廣播頻道
  - 如果是進行廣播，是在各廣播頻道輪流
  - 如果是資料傳輸，則會在連線時，共同約初次傳送的頻道與每次要跳多少個頻道繼續傳輸資料
- 在 Bluetooth 5 以後每個頻道皆可拿來廣播

BLE 的 Channel 分佈



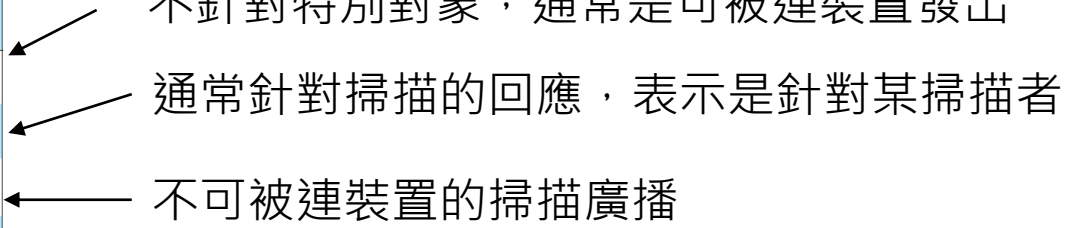
http://www.argenox.com/bluetooth-low-energy-ble-v4-0-development/library/a-ble-advertising-primer/

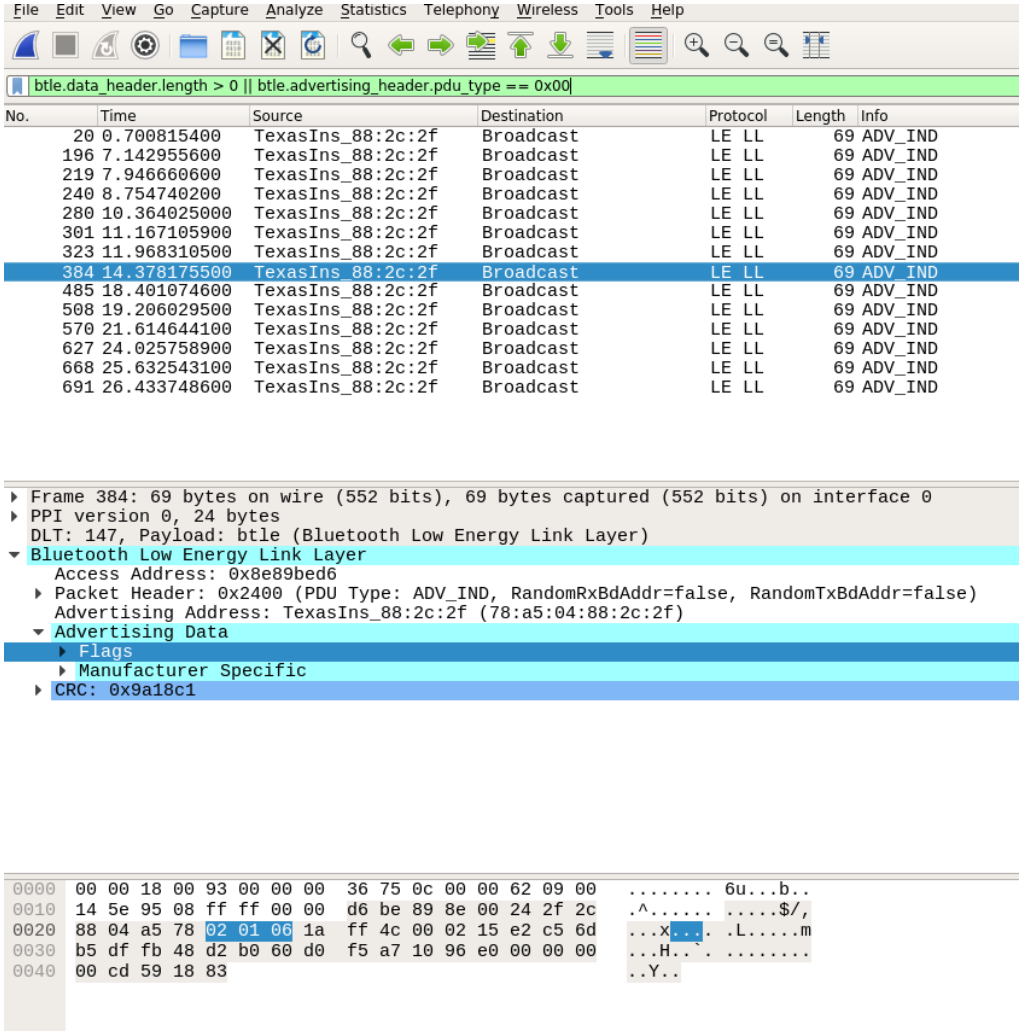
廣播封包格式

LSB MSB

PDU Type (4 bits)	RFU (2 bits)	TxAdd (1 bit)	RxAdd (1 bit)	Length (6 bits)	RFU (2 bits)

PDU Type $b_3b_2b_1b_0$	Packet Name
0000	ADV_IND
0001	ADV_DIRECT_IND
0010	ADV_NONCONN_IND
0011	SCAN_REQ
0100	SCAN_RSP
0101	CONNECT_REQ
0110	ADV_SCAN_IND
0111-1111	Reserved

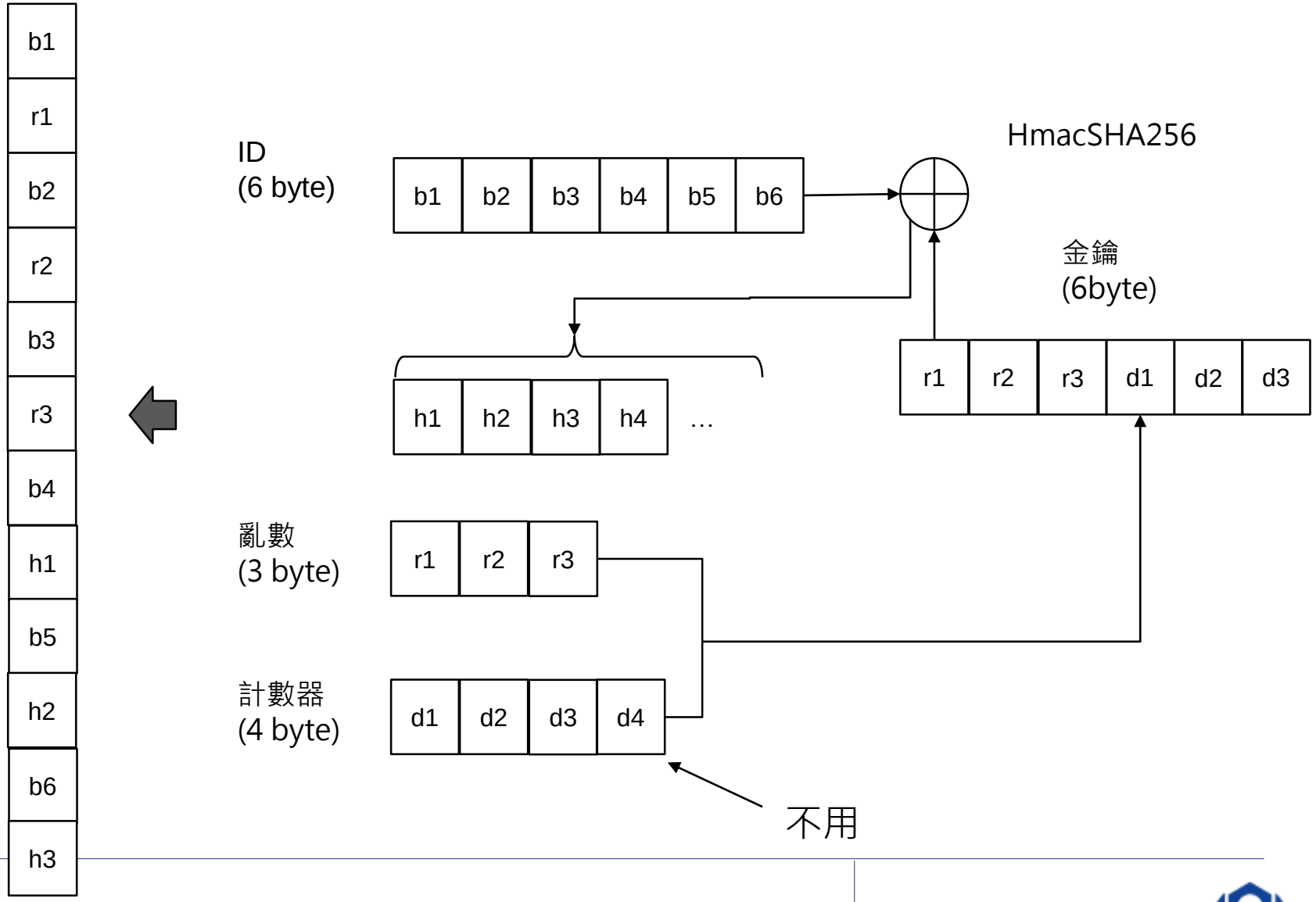



File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
btle.data_header.length > 0 || btle.advertising_header.pdu_type == 0x00
No. Time Source Destination Protocol Length Info
20 0.700815400 TexasIns_88:2c:2f Broadcast LE LL 69 ADV_IND
196 7.142955600 TexasIns_88:2c:2f Broadcast LE LL 69 ADV_IND
219 7.946660600 TexasIns_88:2c:2f Broadcast LE LL 69 ADV_IND
240 8.754740200 TexasIns_88:2c:2f Broadcast LE LL 69 ADV_IND
280 10.364025000 TexasIns_88:2c:2f Broadcast LE LL 69 ADV_IND
301 11.167105900 TexasIns_88:2c:2f Broadcast LE LL 69 ADV_IND
323 11.968310500 TexasIns_88:2c:2f Broadcast LE LL 69 ADV_IND
384 14.378175500 TexasIns_88:2c:2f Broadcast LE LL 69 ADV_IND
485 18.401074600 TexasIns_88:2c:2f Broadcast LE LL 69 ADV_IND
508 19.206029500 TexasIns_88:2c:2f Broadcast LE LL 69 ADV_IND
570 21.614644100 TexasIns_88:2c:2f Broadcast LE LL 69 ADV_IND
627 24.025758900 TexasIns_88:2c:2f Broadcast LE LL 69 ADV_IND
668 25.632543100 TexasIns_88:2c:2f Broadcast LE LL 69 ADV_IND
691 26.433748600 TexasIns_88:2c:2f Broadcast LE LL 69 ADV_IND
Frame 384: 69 bytes on wire (552 bits), 69 bytes captured (552 bits) on interface 0
PPI version 0, 24 bytes
DLT: 147, Payload: btle (Bluetooth Low Energy Link Layer)
Bluetooth Low Energy Link Layer
Access Address: 0x8e89bed6
Packet Header: 0x2400 (PDU Type: ADV_IND, RandomRxBdAddr=false, RandomTxBdAddr=false)
Advertising Address: TexasIns_88:2c:2f (78:a5:04:88:2c:2f)
Advertising Data
Flags
Manufacturer Specific
CRC: 0x9a18c1
0000 00 00 18 00 93 00 00 00 36 75 0c 00 00 62 09 00 .... 6u...b..
0010 14 5e 95 08 ff ff 00 00 d6 be 89 8e 00 24 2f 2c .^...... .....$/,
0020 88 04 a5 78 02 01 06 1a ff 4c 00 02 15 e2 c5 6d ...x.... .L.....m
0030 b5 df fb 48 d2 b0 60 d0 f5 a7 10 96 e0 00 00 00 ...H..`. ....
0040 00 cd 59 18 83 ..Y..

情境一：簡單廣播報到

1. ID, 亂數, 計數器

2. 顯示 ID


b1
r1
b2
r2
b3
r3
b4
h1
b5
h2
b6
h3
ID
(6 byte)
b1
b2
b3
b4
b5
b6
HmacSHA256
金鑰
(6byte)
r1
r2
r3
d1
d2
d3
h1
h2
h3
h4
…
亂數
(3 byte)
r1
r2
r3
計數器
(4 byte)
d1
d2
d3
d4
不用

練習一

- 兩到三人一組 (每組要有筆電)
- 每組領取一指定 ID 與開發板
- 取得封包後，掌握廣播格式，並且以指定 ID 依格式送出訊號
  - 取得指定 ID
  - 選擇 Counter
  - 使用程式語言算出結果
  - 將結果透過 UART 的方式餵給開發版

情境二：用廣播進行挑戰與回應

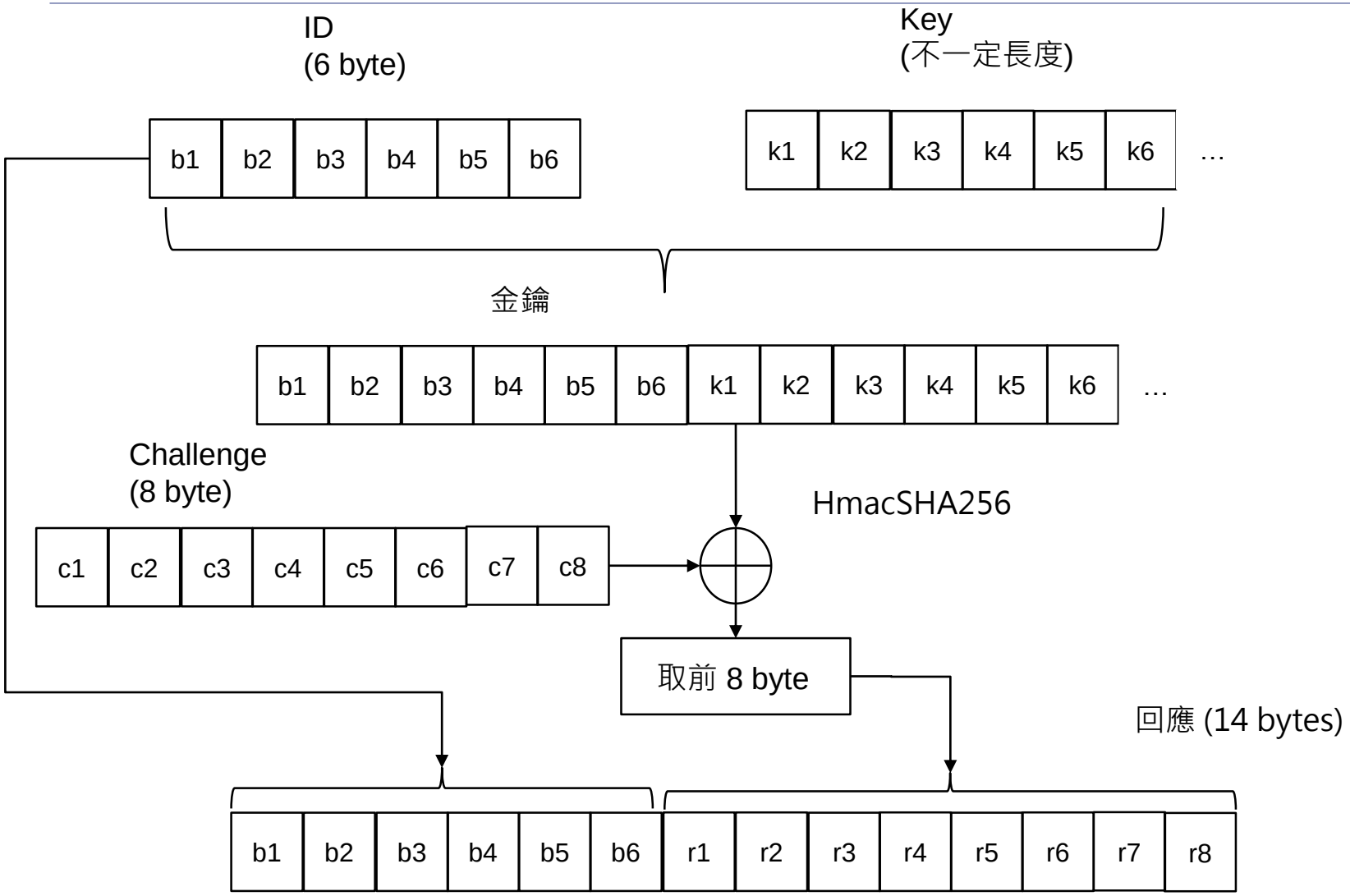
金鑰 K

1. 挑戰 (8 byte 亂數)

2. ID+回應
(6 byte ID + 8 byte 回應)

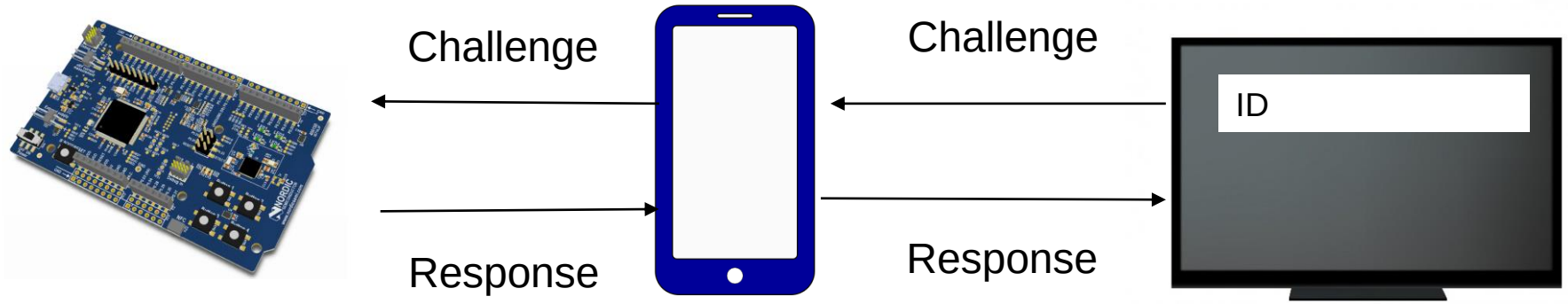
金鑰 K

3. 顯示 ID


ID
(6 byte)
Key
(不一定長度)
b1 b2 b3 b4 b5 b6
k1 k2 k3 k4 k5 k6 …
金鑰
b1 b2 b3 b4 b5 b6 k1 k2 k3 k4 k5 k6 …
Challenge
(8 byte)
HmacSHA256
c1 c2 c3 c4 c5 c6 c7 c8
取前 8 byte
回應 (14 bytes)
b1 b2 b3 b4 b5 b6 r1 r2 r3 r4 r5 r6 r7 r8

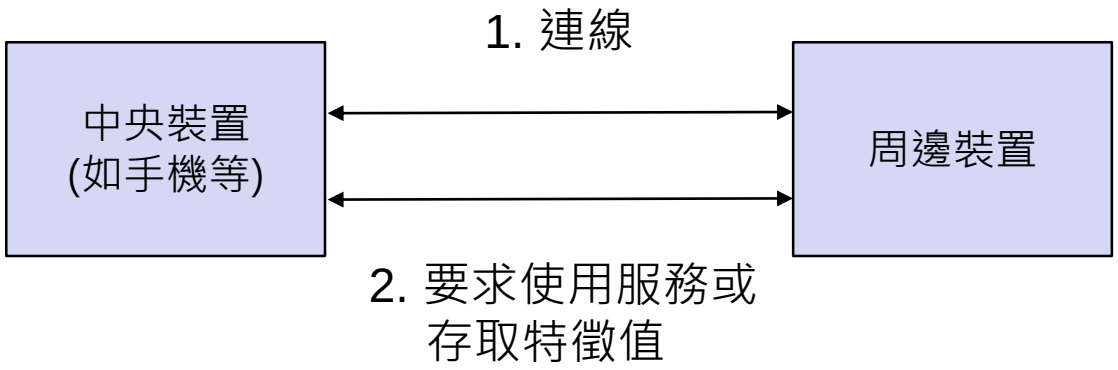
練習二：進行 Relay Attack

- 先取得產生之挑戰，之後送給裝置取得回應
- 並將挑戰傳送給點名系統

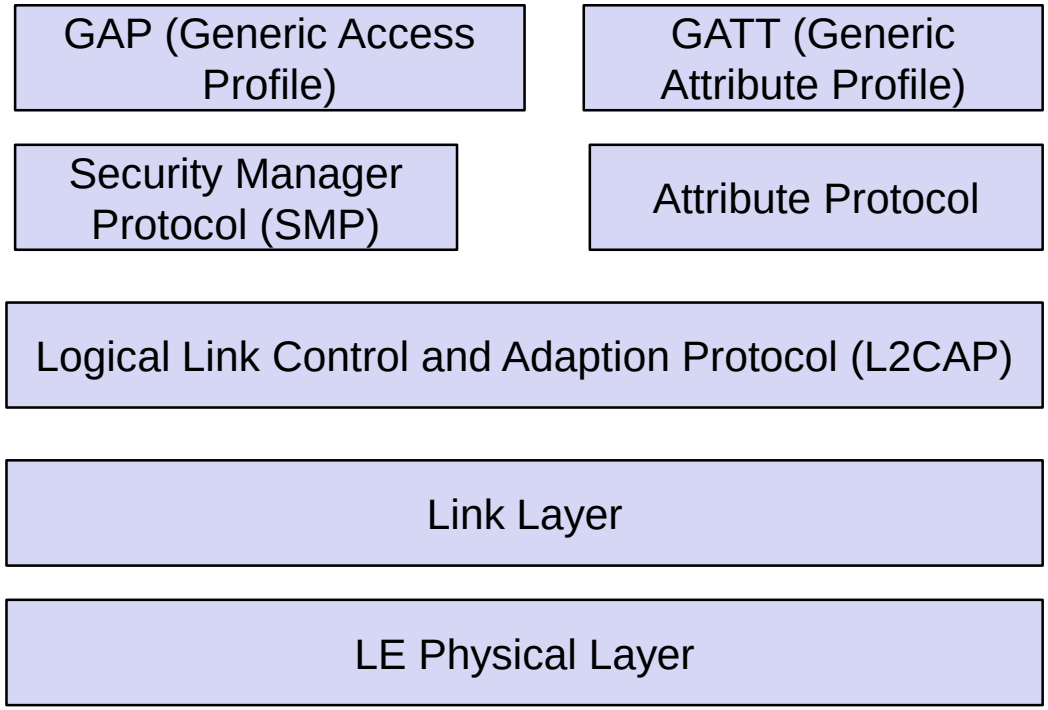


3 BLE 連線與弱點

BLE 連線與服務使用的概念



BLE Protocol Stack

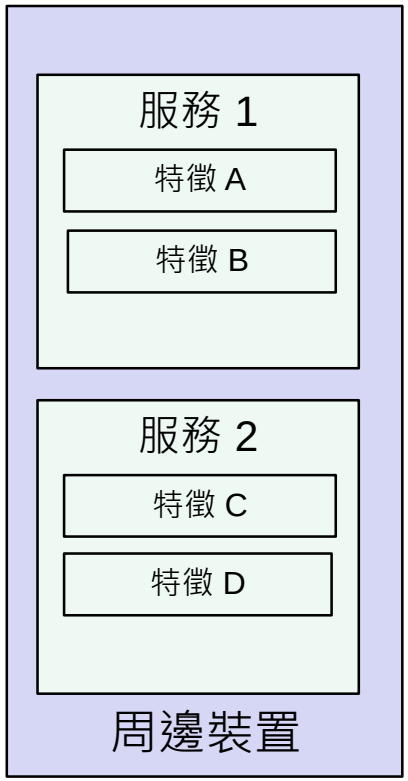


運用 ATT，而在 ATT 所提供的屬性機制上提供服務與服務特徵

提供屬性 (Attribute) 資料，包括：

- 16 bit handle
- 屬性資料類型的 UUID
- 屬性資料的數值

Source: Getting Started with Bluetooth Low Energy - Tools and Techniques for Low-Power Networking



Handle	UUID	Description and value
0x0100	0x2800	Service 1 definition
...	...	Service details
0x0200	0x2800	Service 2 definition
...	...	Service details
0x0300	0x2800	Service 3 definition
...	...	Service details

在兩個服務屬性間就是前一服務的特徵

一個特徵包括：

- 特徵宣告
- 特徵數值
- 特徵描述

Selected AD Type List

	AD Type	Meaning
0x01	Flags	
0x02	16-bit Service UUIDs	More 16-bit UUIDs available
0x03	16-bit Service UUID List	Complete list of 16-bit UUIDs available
0x04	32-bit Service UUIDs	More 32-bit UUIDs available
0x05	32-bit Service UUID List	Complete list of 32-bit UUIDs available
0x06	128-bit Service UUIDs	More 128-bit UUIDs available
0x07	128-bit Service UUID List	Complete list of 128-bit UUIDs available
0x08	Local Name	Shortened local name
0x09	Local Name	Complete local name
0xFF	Manufacture Specific	Company Identifier(2 Octet)

00000000-0000-1000-8000-00805f9b34fb

特徵的組成 -特徵宣告 Characteristic Declaration

Attribute Handle	Attribute Types	Attribute Value			Attribute Permissions
0xNNNN	0x2803–UUID for «Characteristic»	Charac-teristic Properties	Character-istic Value Attribute Handle	Character-istic UUID	Read Only, No Authentication, No Authorization

Attribute Value	Size	Description
Characteristic Properties	1 octets	Bit field of characteristic properties
Characteristic Value Handle	2 octets	Handle of the Attribute containing the value of this characteristic
Characteristic UUID	2 or 16 octets	16-bit Bluetooth UUID or 128-bit UUID for Char-acteristic Value

Properties	Value	Description
Broadcast	0x01	If set, permits broadcasts of the Characteristic Value using Server Characteristic Configuration Descriptor. If set, the Server Characteristic Configuration Descriptor shall exist.
Read	0x02	If set, permits reads of the Characteristic Value using procedures defined in Section 4.8
Write Without Response	0x04	If set, permit writes of the Characteristic Value without response using procedures defined in Section 4.9.1.
Write	0x08	If set, permits writes of the Characteristic Value with response using procedures defined in Section 4.9.3 or Section 4.9.4.
Notify	0x10	If set, permits notifications of a Characteristic Value without acknowledgment using the procedure defined in Section 4.10. If set, the Client Characteristic Configuration Descriptor shall exist.
Indicate	0x20	If set, permits indications of a Characteristic Value with acknowl-edgment using the procedure defined in Section 4.11. If set, the Client Characteristic Configuration Descriptor shall exist.
Authenticated Signed Writes	0x40	If set, permits signed writes to the Characteristic Value using the procedure defined in Section 4.9.2.
Extended Properties	0x80	If set, additional characteristic properties are defined in the Char-acteristic Extended Properties Descriptor defined in Section 3.3.3.1. If set, the Characteristic Extended Properties Descriptor shall exist.

特徵的組成 -特徵數值 Characteristic Value

Attribute Handle	Attribute Type	Attribute Value	Attribute Permissions
0xNNNN	0xuuuu – 16-bit Bluetooth UUID or 128-bit UUID for Characteristic UUID	Characteristic Value	Higher layer profile or implementation specific

Example Attribute Server Attributes

Handle	Attribute Type	Attribute Value
0x0001	«Primary Service»	«GAP Service»
0x0004	«Characteristic»	{0x02, 0x0006, «Device Name»}
0x0006	«Device Name»	“Example Device”
0x0010	«Primary Service»	«GATT Service»
0x0011	«Characteristic»	{0x26, 0x0012, «Service Changed»}
0x0012	«Service Changed»	0x0000, 0x0000
0x0100	«Primary Service»	«Battery State Service»
0x0106	«Characteristic»	{0x02, 0x0110, «Battery State»}
0x0110	«Battery State»	0x04
0x0200	«Primary Service»	«Thermometer Humidity Service»
0x0201	«Include»	{0x0500, 0x0504, «Manufacturer Service»}
0x0202	«Include»	{0x0550,0x0568}
0x0203	«Characteristic»	{0x02, 0x0204, «Temperature»}
0x0204	«Temperature»	0x028A
0x0205	«Characteristic Format»	{0x0E, 0xFE, «degrees Celsius», 0x01, «Outside»}
0x0206	«Characteristic User Description»	“Outside Temperature”
0x0210	«Characteristic»	{0x02, 0x0212, «Relative Humidity»}
0x0212	«Relative Humidity»	0x27
0x0213	«Characteristic Format»	{0x04, 0x00, «Percent», «Bluetooth SIG», «Outside»}
0x0214	«Characteristic User Description»	“Outside Relative Humidity”
0x0280	«Primary Service»	«Weight Service»

#可以帶一下實際錄到的資訊

GATT Specifications

Generic Attributes (GATT) services are collections of characteristics and relationships to other services that encapsulate the behavior of part of a device.

A *GATT profile* describes a use case, roles, and general behaviors based on the GATT functionality, enabling extensive innovation while maintaining full interoperability with other *Bluetooth*® devices.

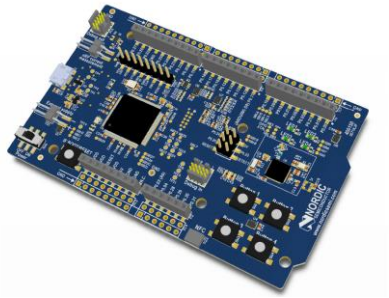
More about GATT

Adopted GATT Profile and Service Specifications

Profile Specification		Version	Status	Date Adopted
ANP	Alert Notification Profile	1.0	Active	13 September 2011
ANS	Alert Notification Service	1.0	Active	13 September 2011
AIOP	Automation IO Profile	1.0	Active	14 July 2015
AIOS	Automation IO Service	1.0	Active	14 July 2015
BAS	Battery Service	1.0	Active	27 December 2011
BCS	Body Composition Service	1.0	Active	21 October 2014
BLP	Blood Pressure Profile	1.0	Active	25 October 2011
BLS	Blood Pressure Service	1.0	Active	25 October 2011
BMS	Bond Management Service	1.0	Active	21 October 2014
CGMP	Continuous Glucose Monitoring Profile	1.0.1	Active	15 December 2015

情境三、連線後使用服務

金鑰 K



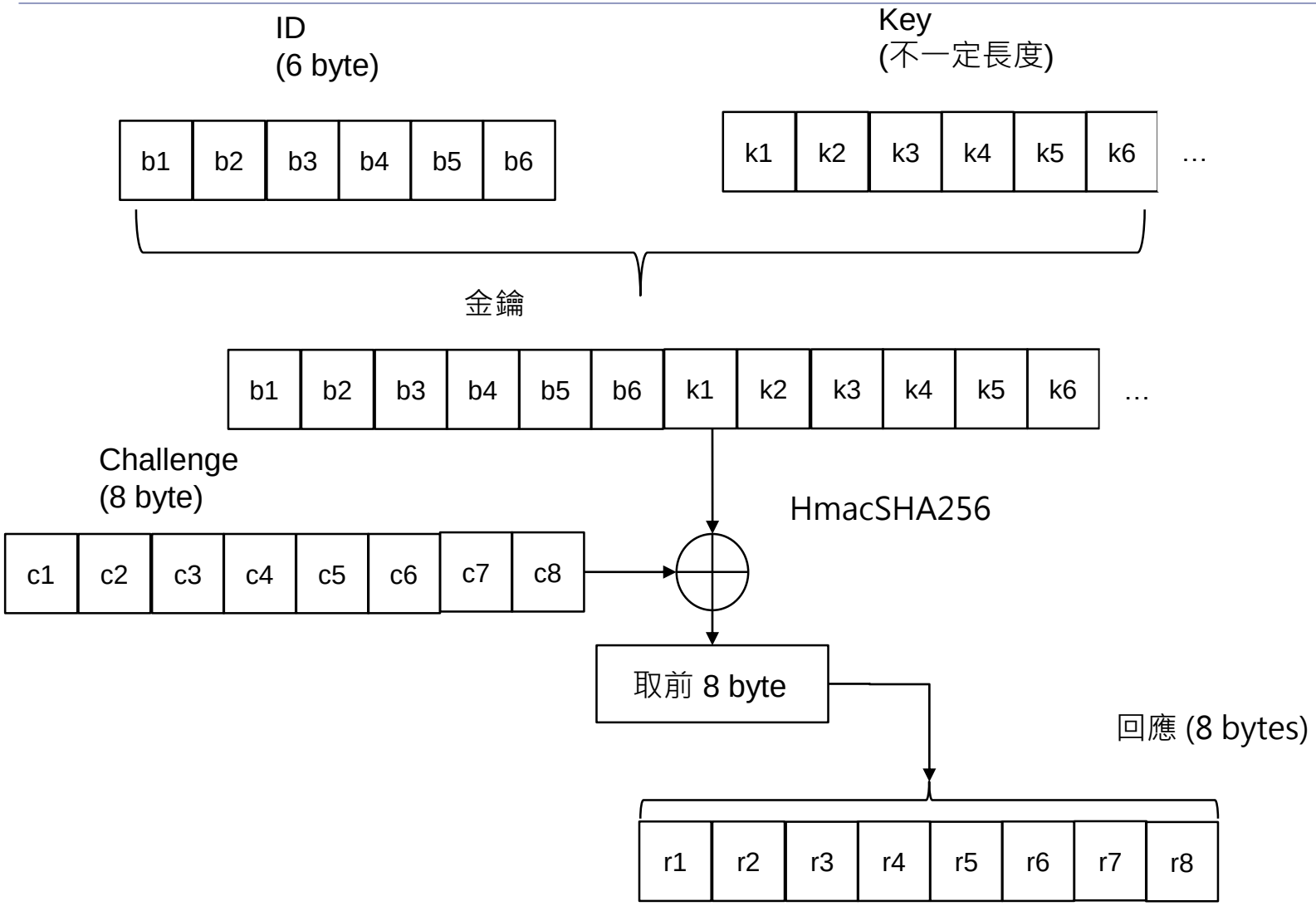
1. 廣播 ID

2. 連線

3. 用寫入特徵值的方式送出挑戰

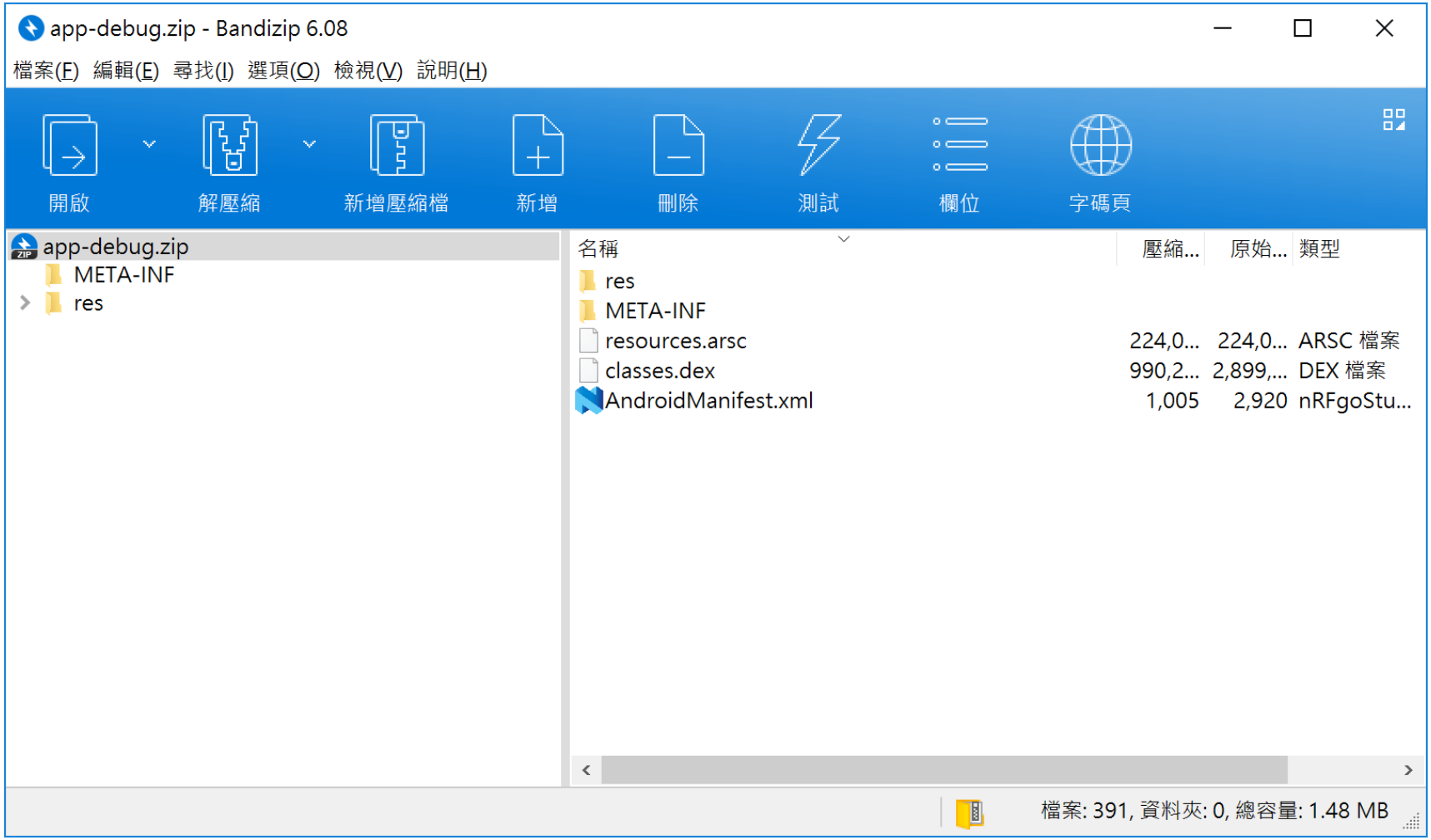
4. 用接受通知 (或讀取特徵值)的方式 取得回應

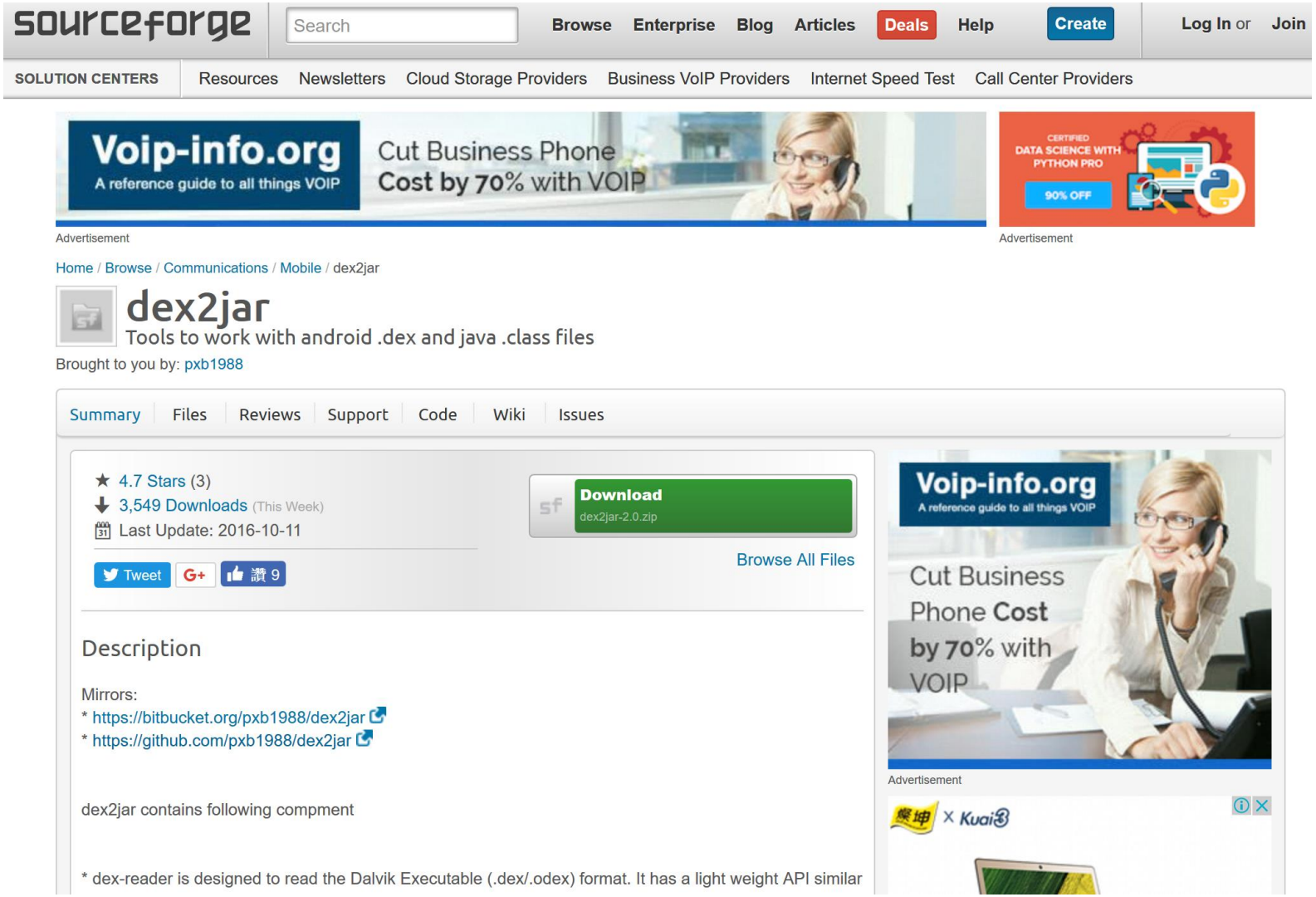
金鑰 K


ID
(6 byte)
b1
b2
b3
b4
b5
b6
Key
(不一定長度)
k1
k2
k3
k4
k5
k6
…
金鑰
b1
b2
b3
b4
b5
b6
k1
k2
k3
k4
k5
k6
…
Challenge
(8 byte)
c1
c2
c3
c4
c5
c6
c7
c8
HmacSHA256
取前 8 byte
回應 (8 bytes)
r1
r2
r3
r4
r5
r6
r7
r8

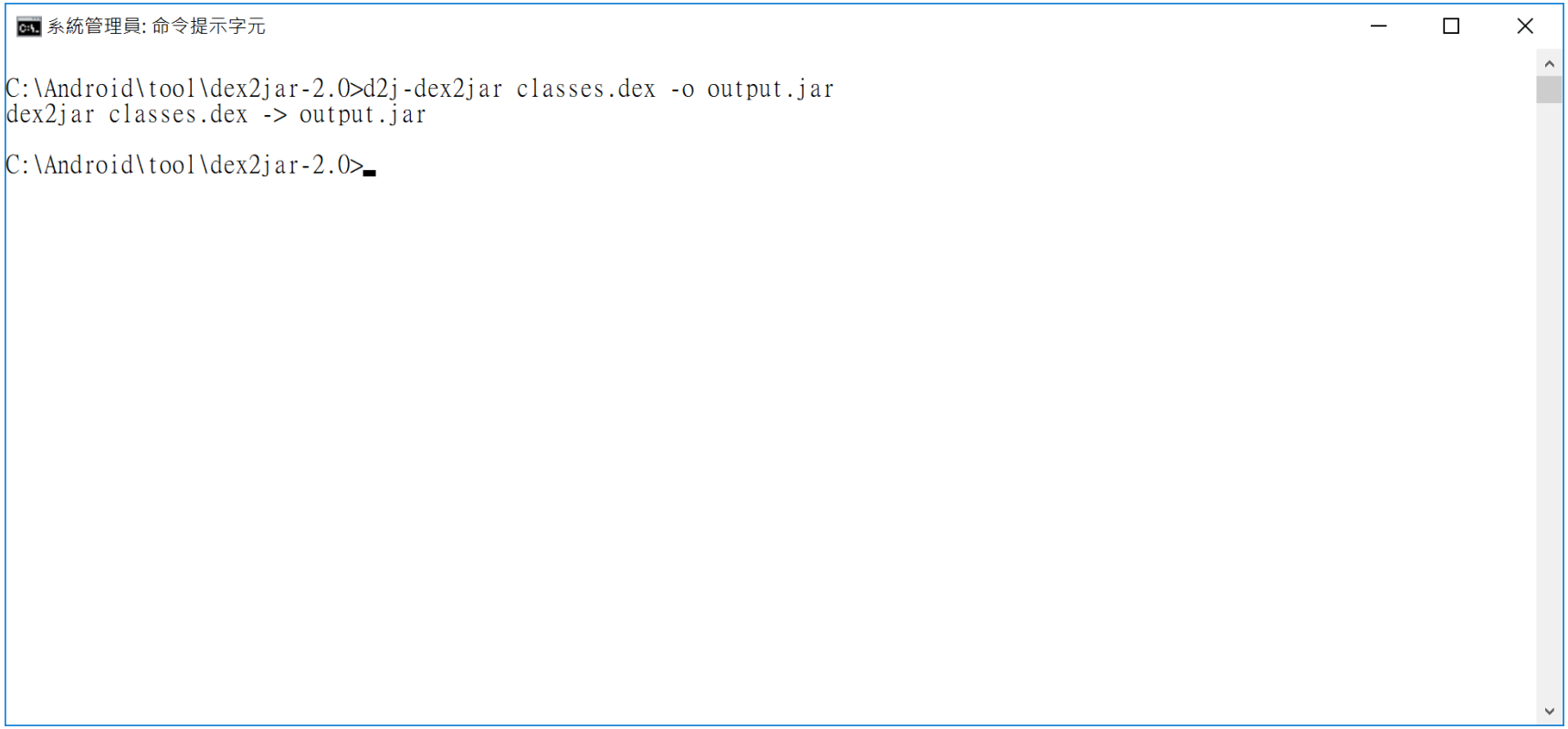
練習三：嘗試取得金鑰

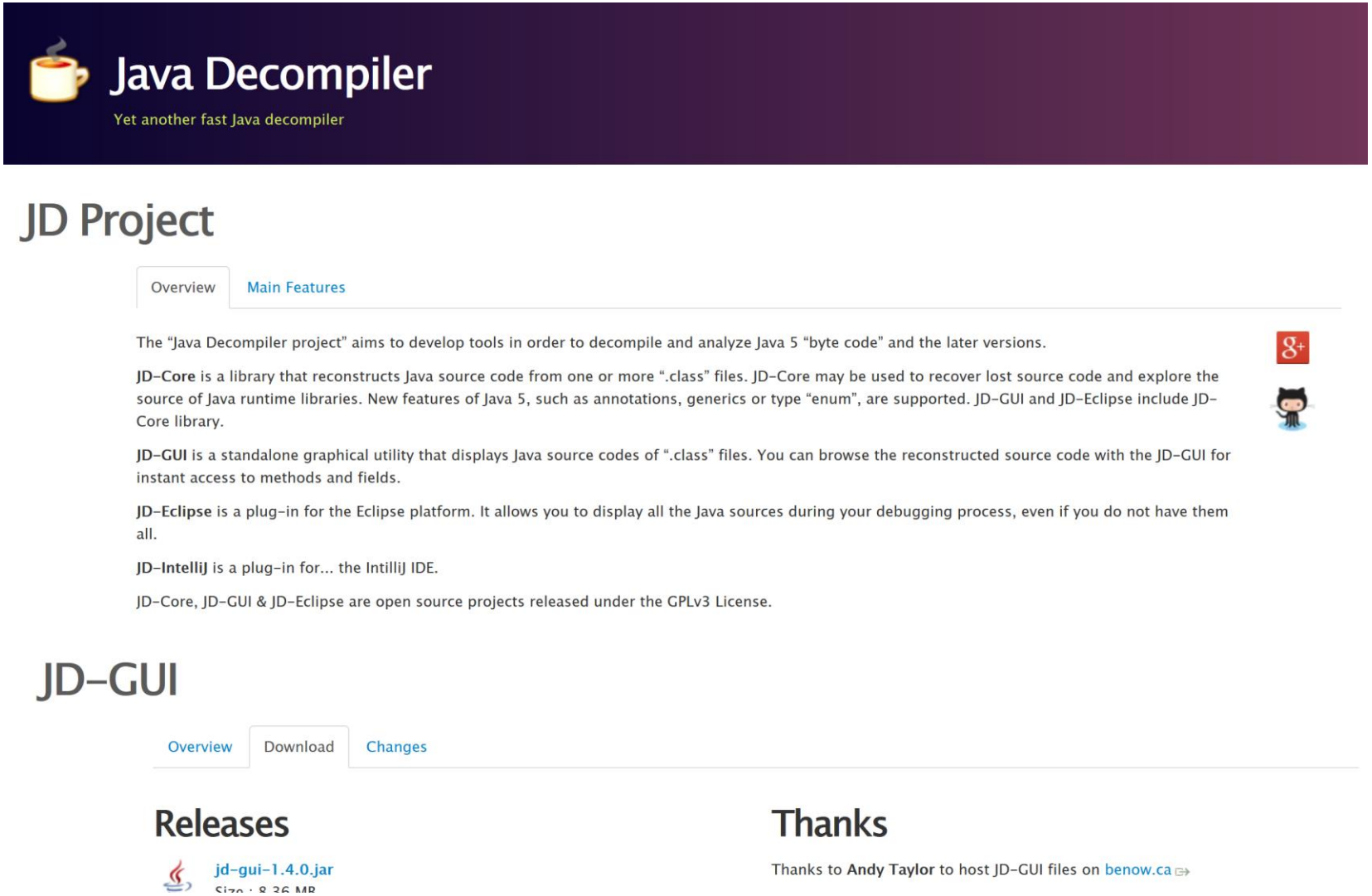
- 透過反組譯 APK 找出金鑰
- 將 ID 與金鑰透過 UART 提供給開發板


app-debug.zip - Bandizip 6.08
檔案(F) 編輯(E) 尋找(I) 選項(O) 檢視(V) 說明(H)
開啟
解壓縮
新增壓縮檔
新增
刪除
測試
欄位
字碼頁
app-debug.zip
META-INF
res
名稱
壓縮...
原始...
類型
res
META-INF
resources.arsc 224,0... 224,0... ARSC 檔案
classes.dex 990,2... 2,899,... DEX 檔案
AndroidManifest.xml 1,005 2,920 nRFgoStu...
檔案: 391, 資料夾: 0, 總容量: 1.48 MB

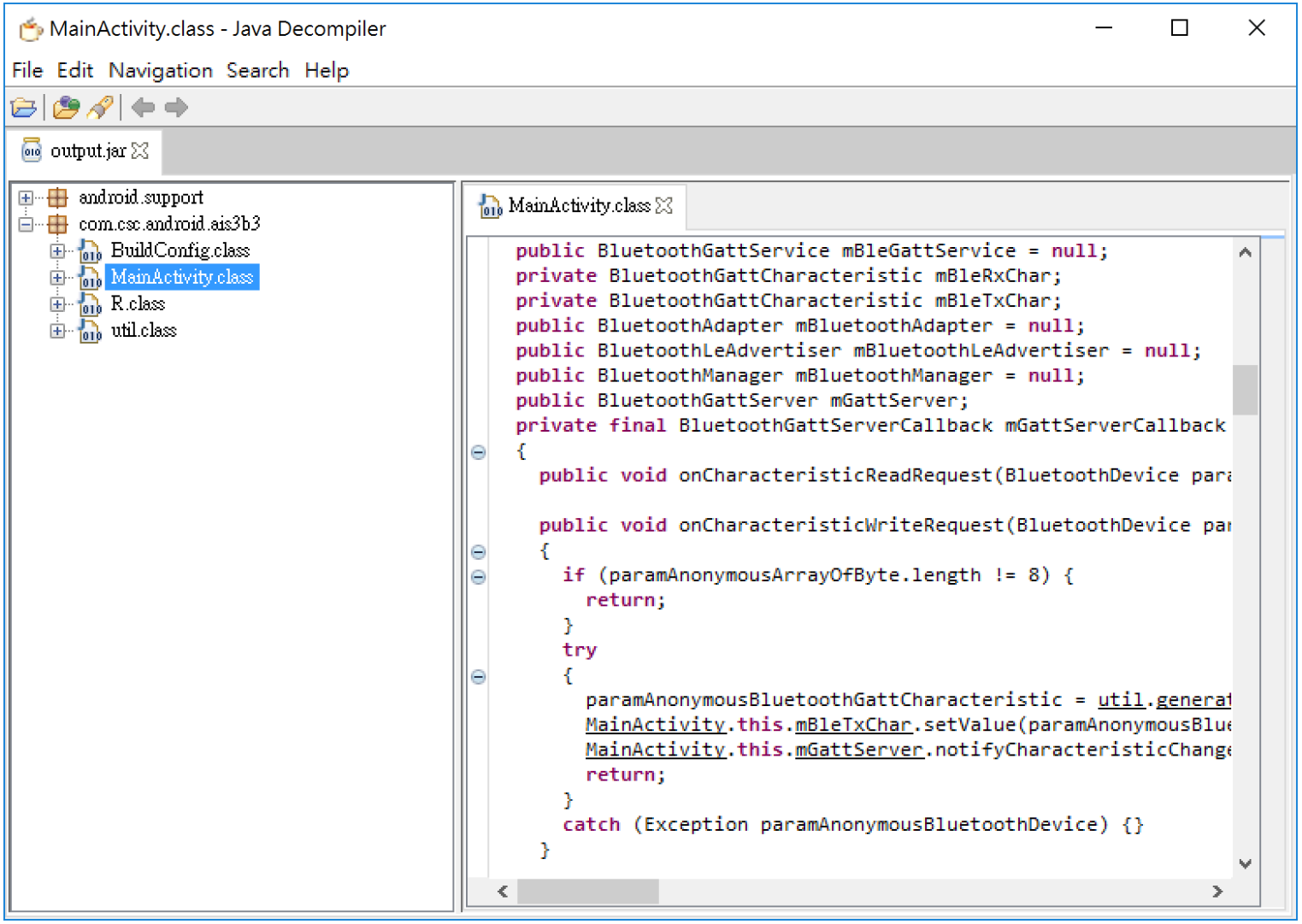


https://sourceforge.net/projects/dex2jar/


系統管理員: 命令提示字元
C:\Android\tool\dex2jar-2.0>d2j-dex2jar classes.dex -o output.jar
dex2jar classes.dex -> output.jar
C:\Android\tool\dex2jar-2.0>



http://jd.benow.ca/


MainActivity.class - Java Decompiler
File Edit Navigation Search Help
output.jar
android.support
com.csc.android.ais3b3
BuildConfig.class
MainActivity.class
R.class
util.class
MainActivity.class
public BluetoothGattService mBleGattService = null;
private BluetoothGattCharacteristic mBleRxChar;
private BluetoothGattCharacteristic mBleTxChar;
public BluetoothAdapter mBluetoothAdapter = null;
public BluetoothLeAdvertiser mBluetoothLeAdvertiser = null;
public BluetoothManager mBluetoothManager = null;
public BluetoothGattServer mGattServer;
private final BluetoothGattServerCallback mGattServerCallback
{
public void onCharacteristicReadRequest(BluetoothDevice para
public void onCharacteristicWriteRequest(BluetoothDevice par
{
if (paramAnonymousArrayOfByte.length != 8) {
return;
}
try
{
paramAnonymousBluetoothGattCharacteristic = util.generat
MainActivity.this.mBleTxChar.setValue(paramAnonymousBlue
MainActivity.this.mGattServer.notifyCharacteristicChange
return;
}
catch (Exception paramAnonymousBluetoothDevice) {}
}

Install Instructions

Quick Check

1. Is at least Java 1.7 installed?
2. Does executing `java -version` on command line / command prompt return 1.7 or greater?
3. If not, please install Java 7+ and make it the default.

Installation for Apktool

- **Windows**:
  1. Download Windows wrapper script (Right click, Save Link As `apktool.bat`)
  2. Download apktool-2 (find newest here)
  3. Rename downloaded jar to `apktool.jar`
  4. Move both files (`apktool.jar` & `apktool.bat`) to your Windows directory (Usually `C://Windows`)
  5. If you do not have access to `C://Windows`, you may place the two files anywhere then add that directory to your Environment Variables System PATH variable.
  6. Try running `apktool` via command prompt
- **Linux**:
  1. Download Linux wrapper script (Right click, Save Link As `apktool`)
  2. Download apktool-2 (find newest here)
  3. Rename downloaded jar to `apktool.jar`
  4. Move both files (`apktool.jar` & `apktool`) to `/usr/local/bin` (root needed)
  5. Make sure both files are executable (`chmod +x`)
  6. Try running `apktool` via cli
- **Mac OS X**:
  1. Download Mac wrapper script (Right click, Save Link As `apktool`)
  2. Download apktool-2 (find newest here)
  3. Rename downloaded jar to `apktool.jar`
  4. Move both files (`apktool.jar` & `apktool`) to `/usr/local/bin` (root needed)
  5. Make sure both files are executable (`chmod +x`)
  6. Try running `apktool` via cli

Note - Wrapper scripts are not needed, but helpful so you don't have to type `java -jar apktool.jar` over and over.

https://ibotpeaches.github.io/Apktool/install/


系統管理員: 命令提示字元
C:\Android\tool\apktool>apktool.bat d app-debug.apk -o output
I: Using Apktool 2.2.4 on app-debug.apk
I: Loading resource table...
I: Decoding AndroidManifest.xml with resources...
I: Loading resource table from file: C:\Users\shich\AppData\Local\apktool\framework\1.apk
I: Regular manifest package...
I: Decoding file-resources...
I: Decoding values */* XMLs...
I: Baksmaling classes.dex...
I: Copying assets and libs...
I: Copying unknown files...
I: Copying original files...
C:\Android\tool\apktool>dir
Volume in drive C is Windows
Volume Serial Number is C6F4-3FE3
Directory of C:\Android\tool\apktool
2017/08/28 下午 01:32 <DIR> .
2017/08/28 下午 01:32 <DIR> ..
2017/08/28 下午 01:05 181 apktool.bat
2017/08/28 下午 01:02 7,254,673 apktool.jar
2017/08/28 下午 01:32 <DIR> app-debug
2017/08/28 下午 01:20 1,558,559 app-debug.apk
2017/08/28 下午 01:32 <DIR> output
3 File(s) 8,813,413 bytes
4 Dir(s) 184,450,662,400 bytes free

BLE 的安全機制

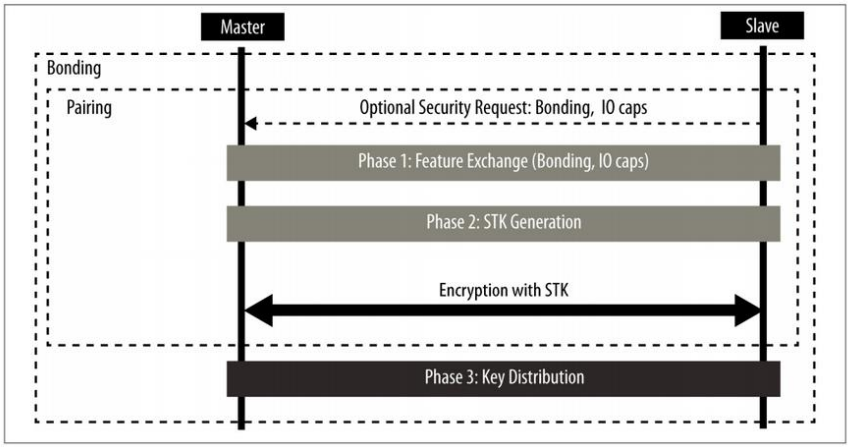
- 基本上有兩種 Security Mode：

基本上這些部分都需要配對

- Security Mode 1：
  - 1. No security (No authentication and no encryption)
  - 2. Unauthenticated pairing with encryption
  - 3. Authenticated pairing with encryption
  - 4. Authenticated LE Secure Connections pairing with encryption using a 128-bit strength encryption key.
- Security Mode 2:
  - 1. Unauthenticated pairing with data signing
  - 2. Authenticated pairing with data signing

之前我們使用的 BLE 服務，原則上是連上了就可以用

- 但如果要考量到安全，一般要做配對與綁定
- 這裡面有兩個主要的動作：
  - 配對 (Pairing)：特別要求連線要符合某些特性
  - 綁定 (Bonding)：交換金鑰以特別針對綁定的連線使用
- 在那之前我們看有沒有工具可以幫我們分析 BLE 封包

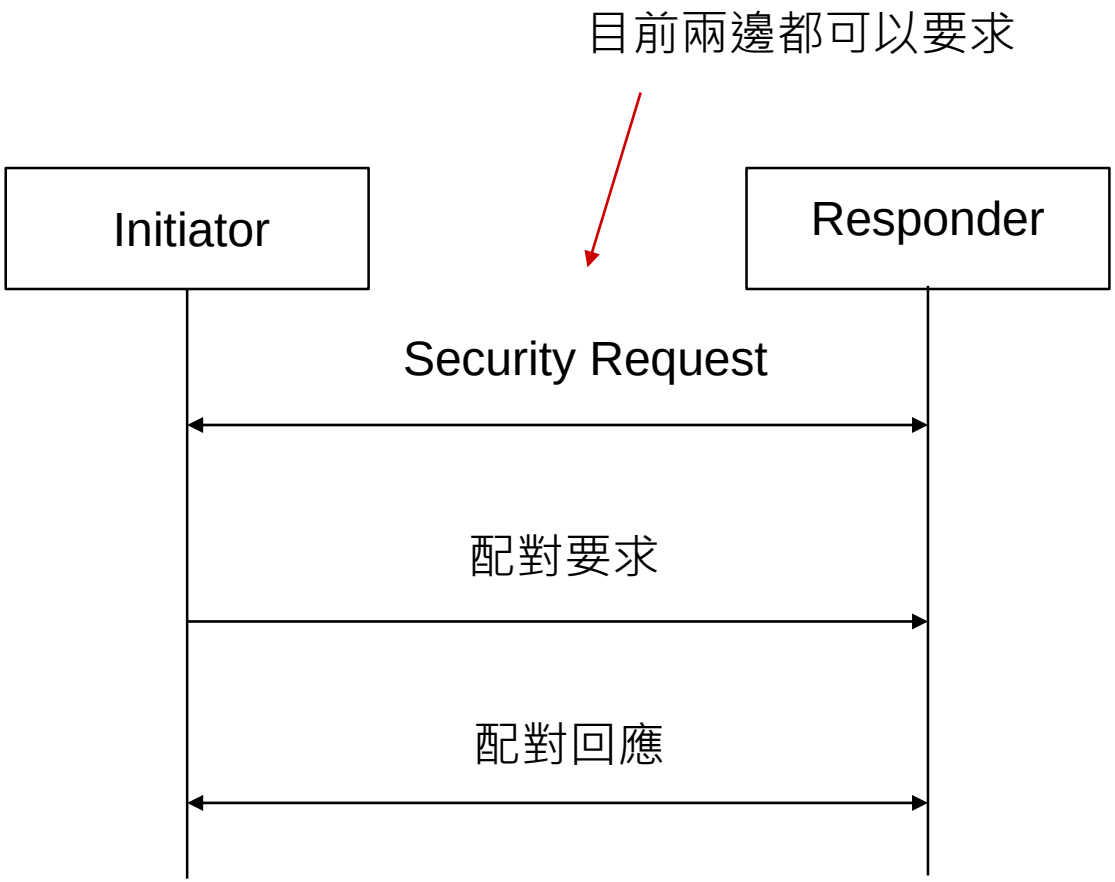


Source: Getting Start With Bluetooth Low Energy

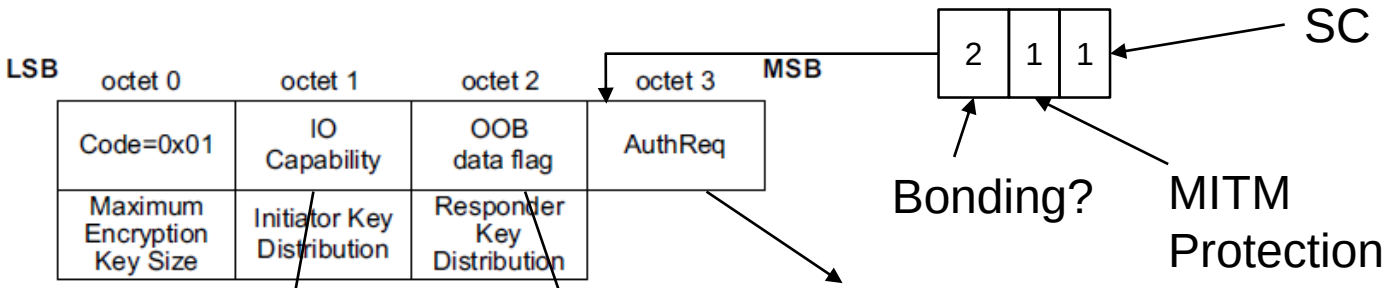
藍牙在配對上分成傳統配對與安全配對 (4.2 之後加入)

- 四種鑑別方式 (傳統配對只支援後三種)
  - Numeric Comparison
    - 裝置兩邊顯示亂數產生的六位數數字
  - Just Work:
    - 互相透過變數的交換，以便產生安全連線的金鑰，以便之後交換資訊
    - 交換變數的過程中，因為變數沒有被保護，因此有可能會被中間人攻擊
  - Passkey Display
    - 在螢幕上顯示密碼，或要求對方輸入密碼
  - Out of Bound (OOB)：透過另一個 Channel 交換認證金鑰

在第一個 Phase 的主要過程



Pairing Request



- LE Secure Connections pairing
- Authenticated MITM protection
- Unauthenticated no MITM protection
- No security requirements

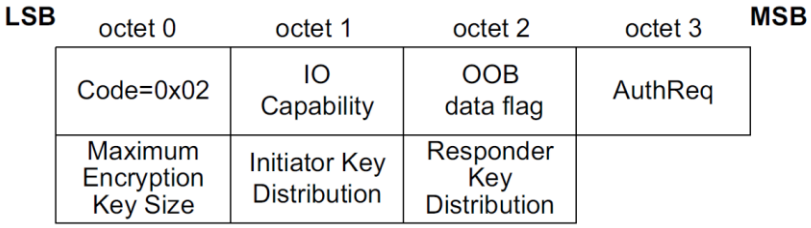
Value	Description
0x00	DisplayOnly
0x01	DisplayYesNo
0x02	KeyboardOnly
0x03	NoInputNoOutput
0x04	KeyboardDisplay
0x05-0xFF	Reserved

		Local output capacity	
		No output	**Numeric output**
Local input capacity	**No input**	NoInputNoOutput	DisplayOnly
	Yes/No	NoInputNoOutput[1]	DisplayYesNo
	Keyboard	KeyboardOnly	KeyboardDisplay

Value	Description
0x00	OOB Authentication data not present
0x01	OOB Authentication data from remote device present
0x02-0xFF	Reserved

至少有兩個按鈕可以選 Yes/No，因此要搭配顯示

Pairing Response 原則上格式和 Request 一樣



但其實不是你要什麼配對方式都行，要看人家支不支援

決定配對演算法的方法 (LE Legacy Pairing)

		Initiator			
		OOB Set	**OOB Not Set**	**MITM Set**	**MITM Not Set**
Responder	**OOB Set**	Use OOB	Check MITM		
	OOB Not Set	Check MITM	Check MITM		
	MITM Set			Use IO Capabilities	Use IO Capabilities
	MITM Not Set			Use IO Capabilities	Use Just Works

萬一兩邊要求不同 Pairing 演算法的解決方法 (LE Secure Connections Pairing)

		Initiator			
		OOB Set	**OOB Not Set**	**MITM Set**	**MITM Not Set**
Responder	**OOB Set**	Use OOB	Use OOB		
	OOB Not Set	Use OOB	Check MITM		
	MITM Set			Use IO Capabilities	Use IO Capabilities
	MITM Not Set			Use IO Capabilities	Use Just Works

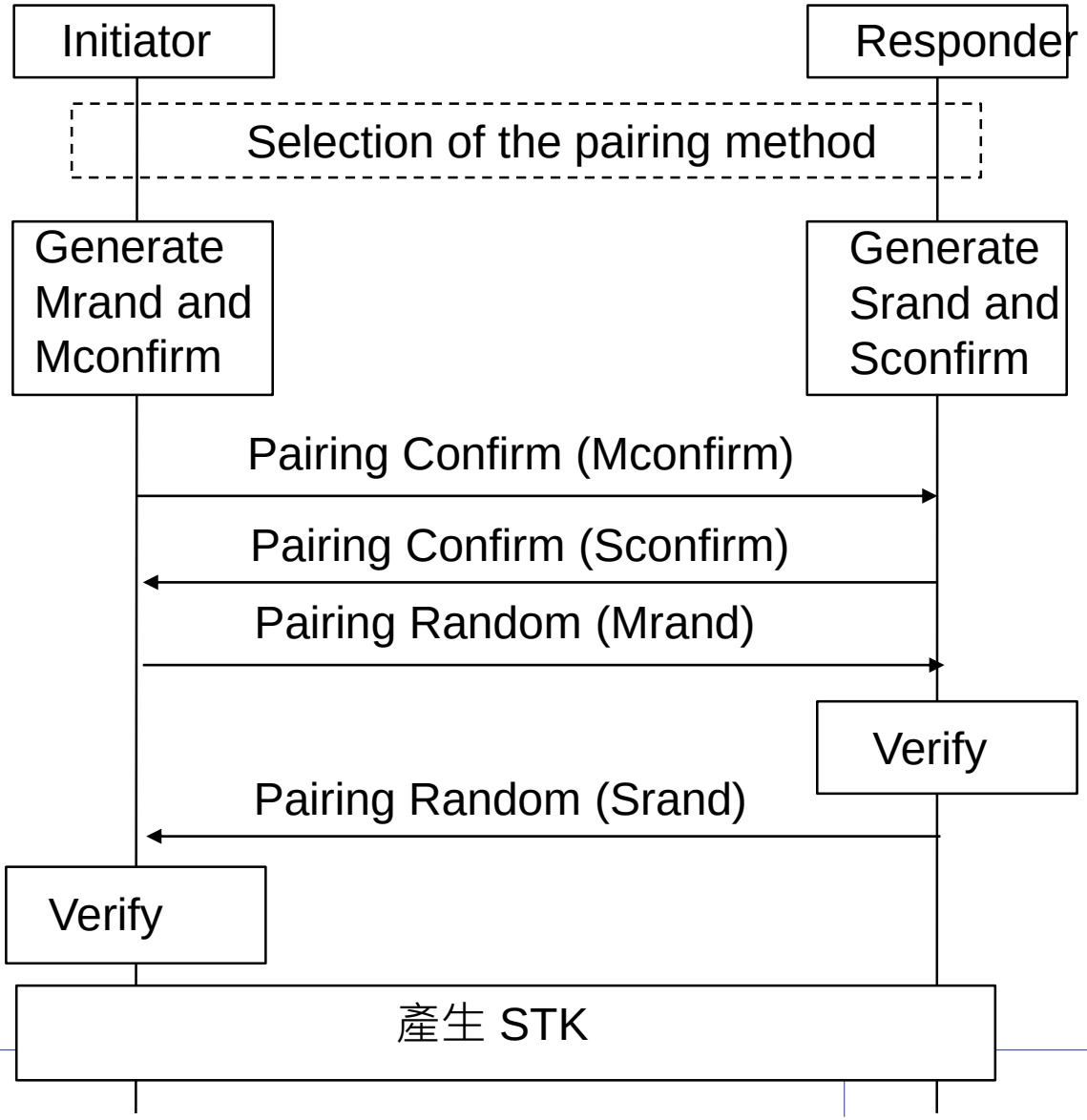
依 I/O 能力而可以做的選擇

	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, initiator inputs Authenti-cated	Just Works Unauthenti-cated	Passkey Entry: responder displays, initiator inputs Authenti-cated
Display YesNo	Just Works Unauthenti-cated	Just Works (For LE Legacy Pairing) Unauthenti-cated	Passkey Entry: responder displays, initiator inputs Authenti-cated	Just Works Unauthenti-cated	Passkey Entry (For LE Legacy Pairing): responder displays, initiator inputs Authenti-cated
		Numeric Comparison (For LE Secure Con-nections) Authenti-cated			Numeric Comparison (For LE Secure Con-nections) Authenti-cated

依 I/O 能力而可以做的選擇 (續)

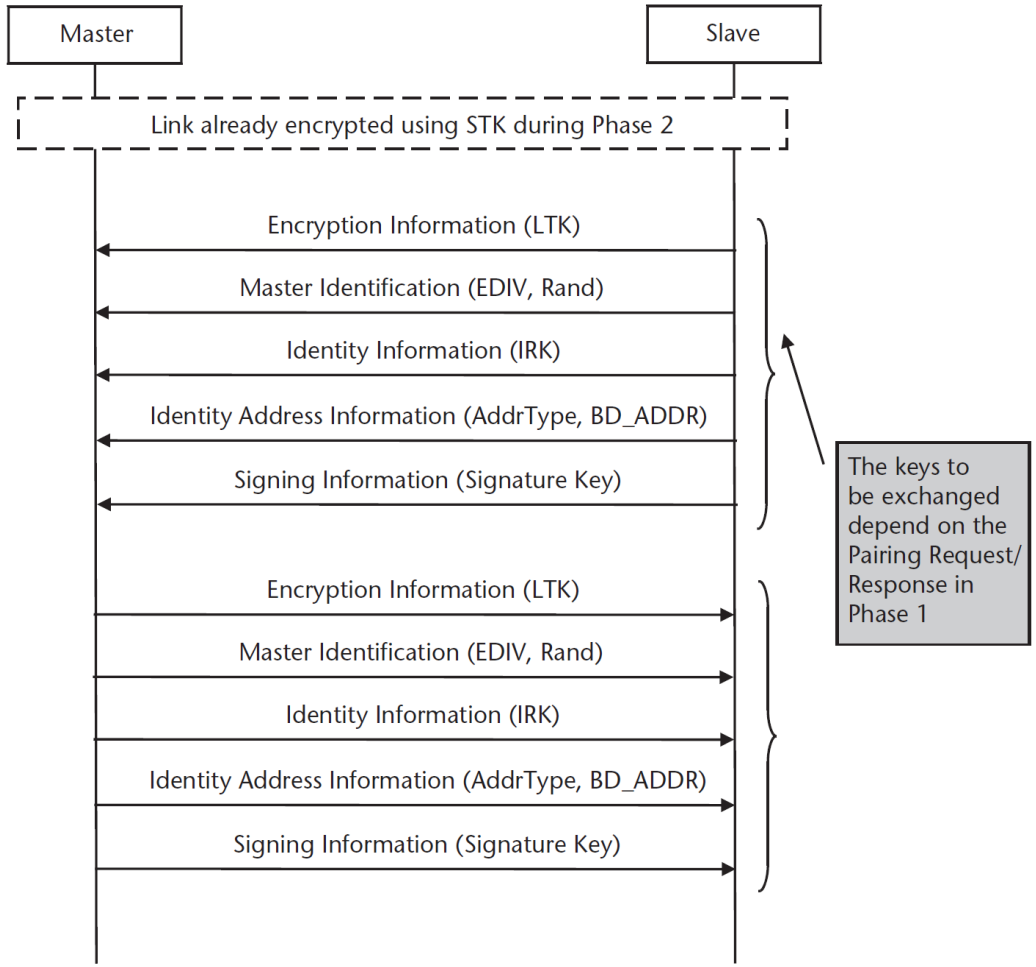
	Initiator				
Responder	**DisplayOnly**	**Display YesNo**	**Keyboard Only**	**NoInput NoOutput**	**Keyboard Display**
Keyboard Only	Passkey Entry: initiator displays, responder inputs Authenticated	Passkey Entry: initiator displays, responder inputs Authenticated	Passkey Entry: initiator and responder inputs Authenticated	Just Works Unauthenticated	Passkey Entry: initiator displays, responder inputs Authenticated
NoInput NoOutput	Just Works Unauthenticated	Just Works Unauthenticated	Just Works Unauthenticated	Just Works Unauthenticated	Just Works Unauthenticated
Keyboard Display	Passkey Entry: initiator displays, responder inputs Authenticated	Passkey Entry (For LE Legacy Pairing): initiator displays, responder inputs Authenticated	Passkey Entry: responder displays, initiator inputs Authenticated	Just Works Unauthenticated	Passkey Entry (For LE Legacy Pairing): initiator displays, responder inputs Authenticated
		Numeric Comparison (For LE Secure Connections) Authenticated			Numeric Comparison (For LE Secure Connections) Authenticated

第二階段的主要動作


Initiator
Responder
Selection of the pairing method
Generate Mrand and Mconfirm
Generate Srand and Sconfirm
Pairing Confirm (Mconfirm)
Pairing Confirm (Sconfirm)
Pairing Random (Mrand)
Verify
Pairing Random (Srand)
Verify
產生 STK

總之配對後會有以下這幾把金鑰完成交換

- Identity Resolving Key (IRK) is a 128-bit key used to generate and resolve random addresses.
- Connection Signature Resolving Key (CSRK) is a 128-bit key used to sign data and verify signatures on the receiving device.
- Long Term Key (LTK) is a 128-bit key used to generate the contributory session key for an encrypted connection.
- Encrypted Diversifier (EDIV) (16-bit) and Random Number (Rand) (64-bit) stored valued used to identify the LTK distributed during LE legacy pairing.



前面交換的金鑰與安全模式間的關係

- Security Mode 1 :
  - 1. No security (No authentication and no encryption)
  - 2. Unauthenticated pairing with encryption
  - 3. Authenticated pairing with encryption
  - 4. Authenticated LE Secure Connections pairing with encryption using a 128-bit strength encryption key.

- Security Mode 2:
  - 1. Unauthenticated pairing with data signing
  - 2. Authenticated pairing with data signing

原則上 Pairing Confirm 的內容是 16 bytes = 128 bits

Code=0x03	Confirm Value
Confirm Value	
Confirm Value	
Confirm Value	
Confirm Value	

Mconfirm = *c1*(*TK*, *Mrand*,
Pairing Request command, Pairing Response command,
initiating device address type, initiating device address,
responding device address type, responding device address)

Sconfirm = *c1*(*TK*, *Srand*,
Pairing Request command, Pairing Response command,
initiating device address type, initiating device address,
responding device address type, responding device address)

c1 (k, r, preq, pres, iat, rat, ia, ra) = e(k, e(k, r XOR p1) XOR p2)

由 rat 補 7 個 0 產生

p1 = pres || preq || rat' || iat'

p2 = padding || ia || ra

不到 128 bits，剩下全補 0

Code=0x04	Random value
Random value	
Random value	
Random value	
Random value	

STK = *s1*(*TK*, *Srand*, *Mrand*)

s1(k, *r1*, *r2*) = e(k, r')

會從 Srand 取前 64bit+Mrand 的前 64bit 產生 r'

e目前使用 AES 128

如果這樣，有沒想過少了甚麼資訊沒提供？

- 如果不是 LE Secure Connection Pairing，我們把它叫做傳統做法：
  - 此時 Just Work 作法上的 TK 為 128 bits 都是 0
  - 如果是要求使用者輸入 Passkey 的，目前 Passkey 為 6 碼數字
    - 000000~999999
  - 如果 passkey = 019655 則
    - TK = 0x00000000000000000000000004CC7
- 所以這表示？
  - 只要不是用 LE Secure Connection Pairing
    - 如果是 Just Work，可以在收到亂數後，直接用 TK 的破 STK
    - 如果是 Passkey，可以用暴力破解
- 如果是 OOB，基本上就比較無法使用暴力破解法去猜，但是如果攔到 TK，一樣可以取得 STK

Paper 有提到相關的弱點

Bluetooth: With Low Energy comes Low Security

Mike Ryan
iSEC Partners

Abstract

We discuss our tools and techniques to monitor and inject packets in Bluetooth Low Energy. Also known as BTLE or Bluetooth Smart, it is found in recent high-end smartphones, sports devices, sensors, and will soon appear in many medical devices. We show that we can effectively render useless the encryption of any Bluetooth Low Energy link.

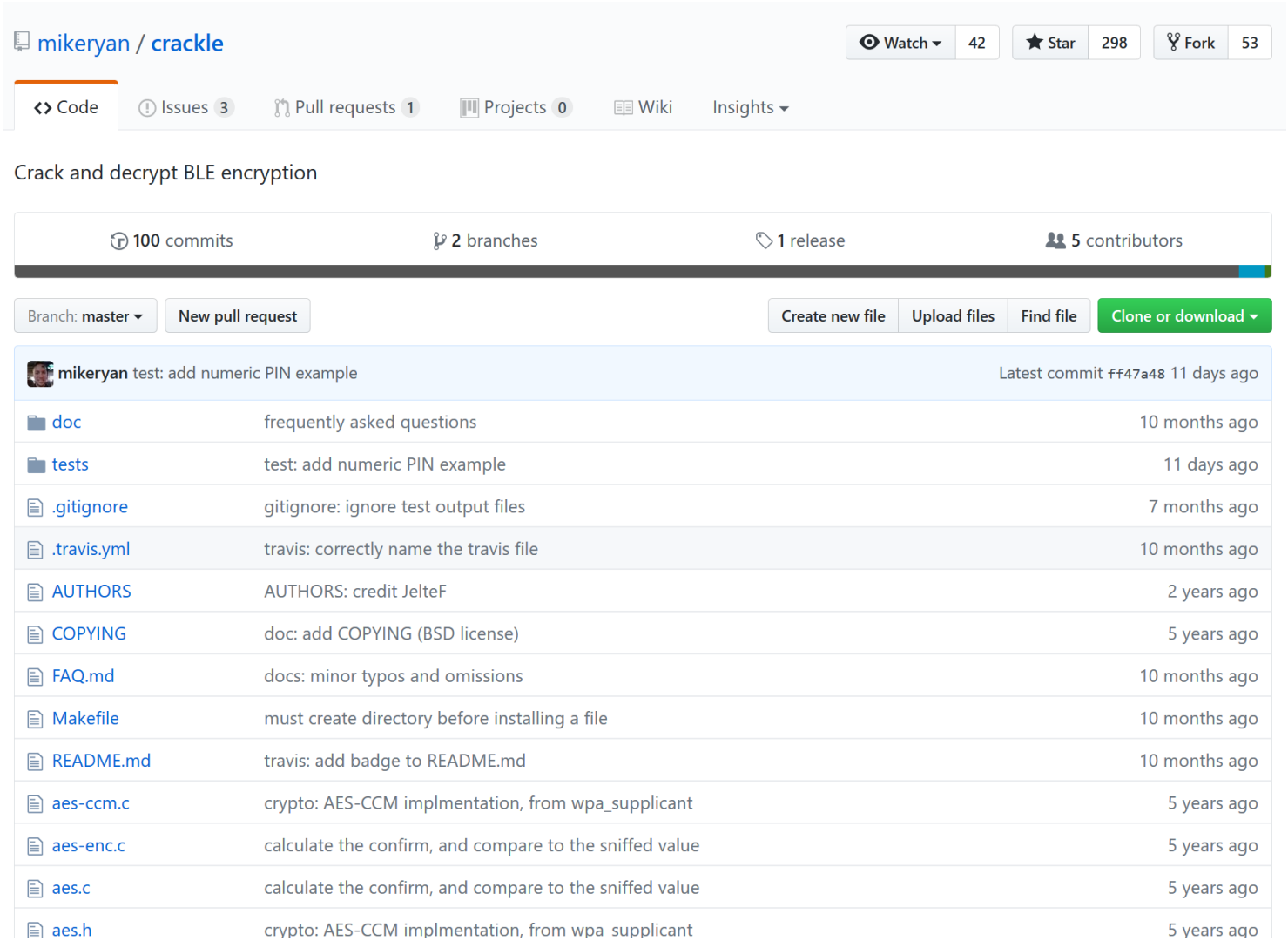
1 Introduction

Bluetooth Low Energy, also known as BTLE or Bluetooth Smart, is a new modulation mode and link layer packet format targeting low power embedded devices. It is typically found in recent high-end smartphones, sports devices, various sensors, and will soon appear in many form. Leveraging the power of the platform we are able to obtain the parameters required to recover encryption keys by using brute force search over a very small keyspace.

2 Bluetooth Low Energy

Bluetooth is a short range connectivity protocol used in 9 billion devices. The number of devices integrating BTLE is expected to grow by 2.9 billion devices per year by 2016 [2].

Bluetooth Low Energy, defined in the Bluetooth Core Spec 4.0 [4], is a wireless protocol operating in the unlicensed 2.4 GHz band. While it operates in the same frequency range as other Bluetooth technologies, its operation at the PHY and link layers is incompatible. At the PHY layer BTLE uses Gaussian Frequency Shift Keying



https://github.com/mikeryan/crackle

mikeryan / **crackle**

Watch 42 | Star 298 | Fork 53

Code | Issues 3 | Pull requests 1 | Projects 0 | Wiki | Insights

Branch: master | crackle / tests / 01_crack / | Create new file | Upload files | Find file | History

mikeryan bugfix: decrypt packets of length 5 or greater ... Latest commit `10c6d65` on 15 Jan

..		
out	bugfix: decrypt packets of length 5 or greater	7 months ago
README.md	tests: add test harness and basic test cases	10 months ago
pairing_and_ltk_exchange.pcap	tests: add test harness and basic test cases	10 months ago

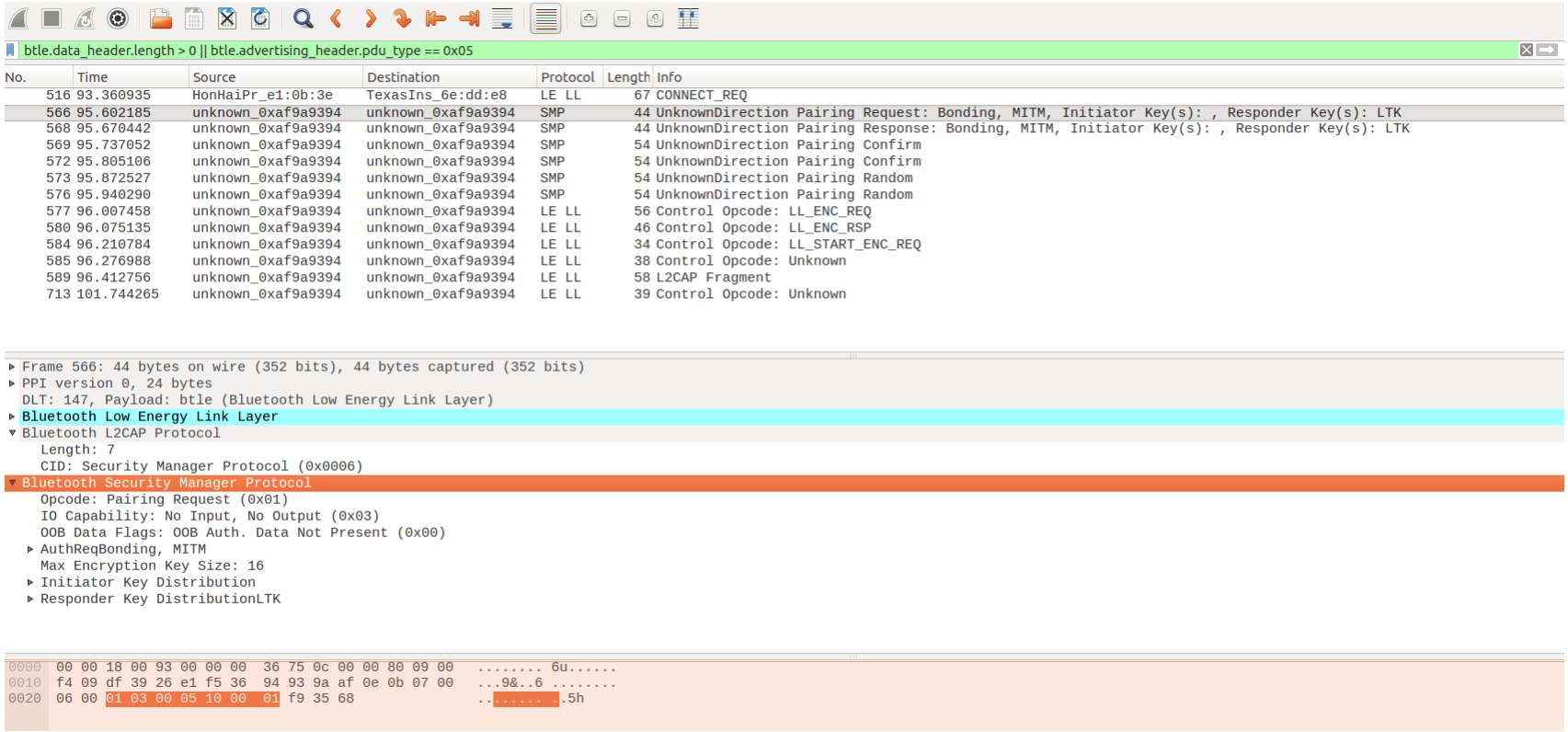
README.md

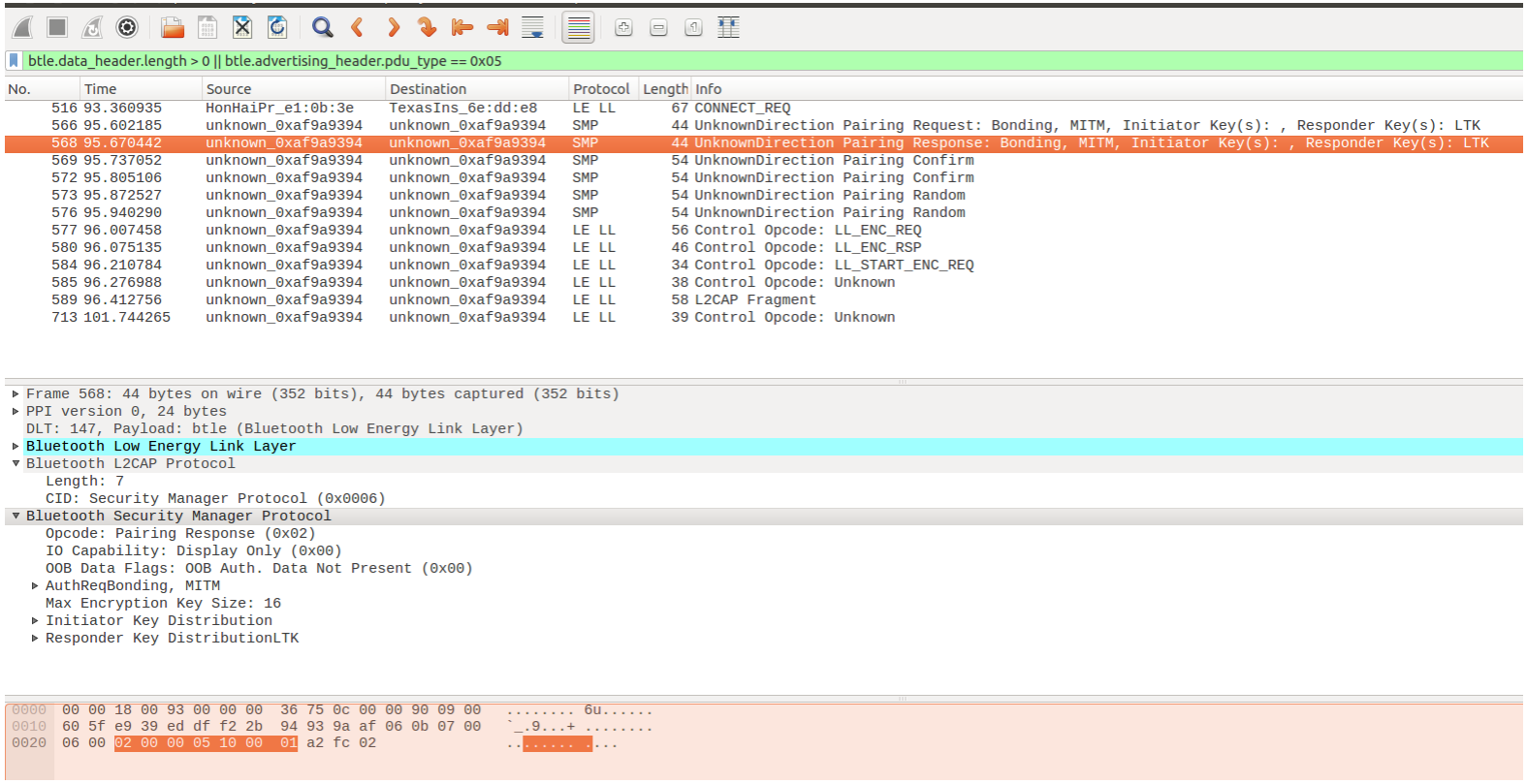
This test validates that crackle's TK cracking with all pairing packets present functions as expected. If the test succeeds, crackle will discover that the TK is 000000 (Just Works pairing) and decrypt the encrypted packets within the PCAP. crackle will extract the LTK that is exchanged after encryption is established and print it to stdout.

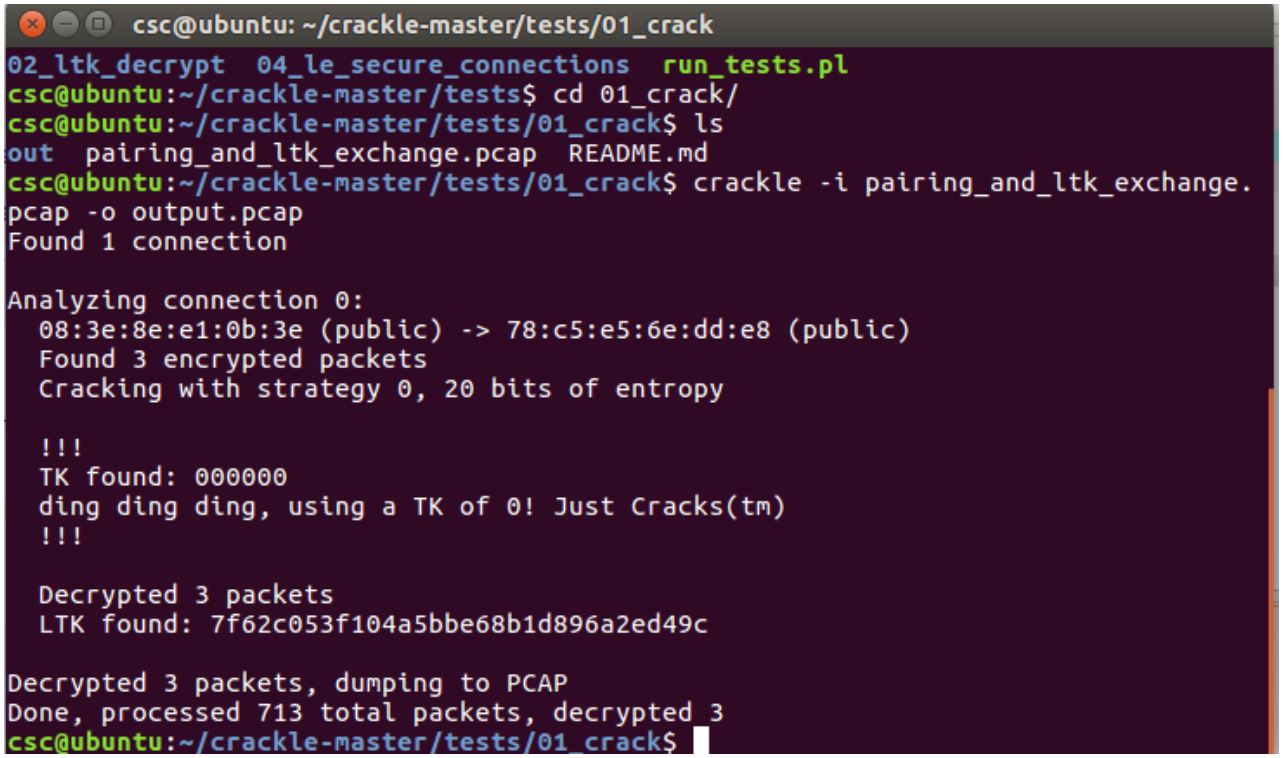
To run this test manually:

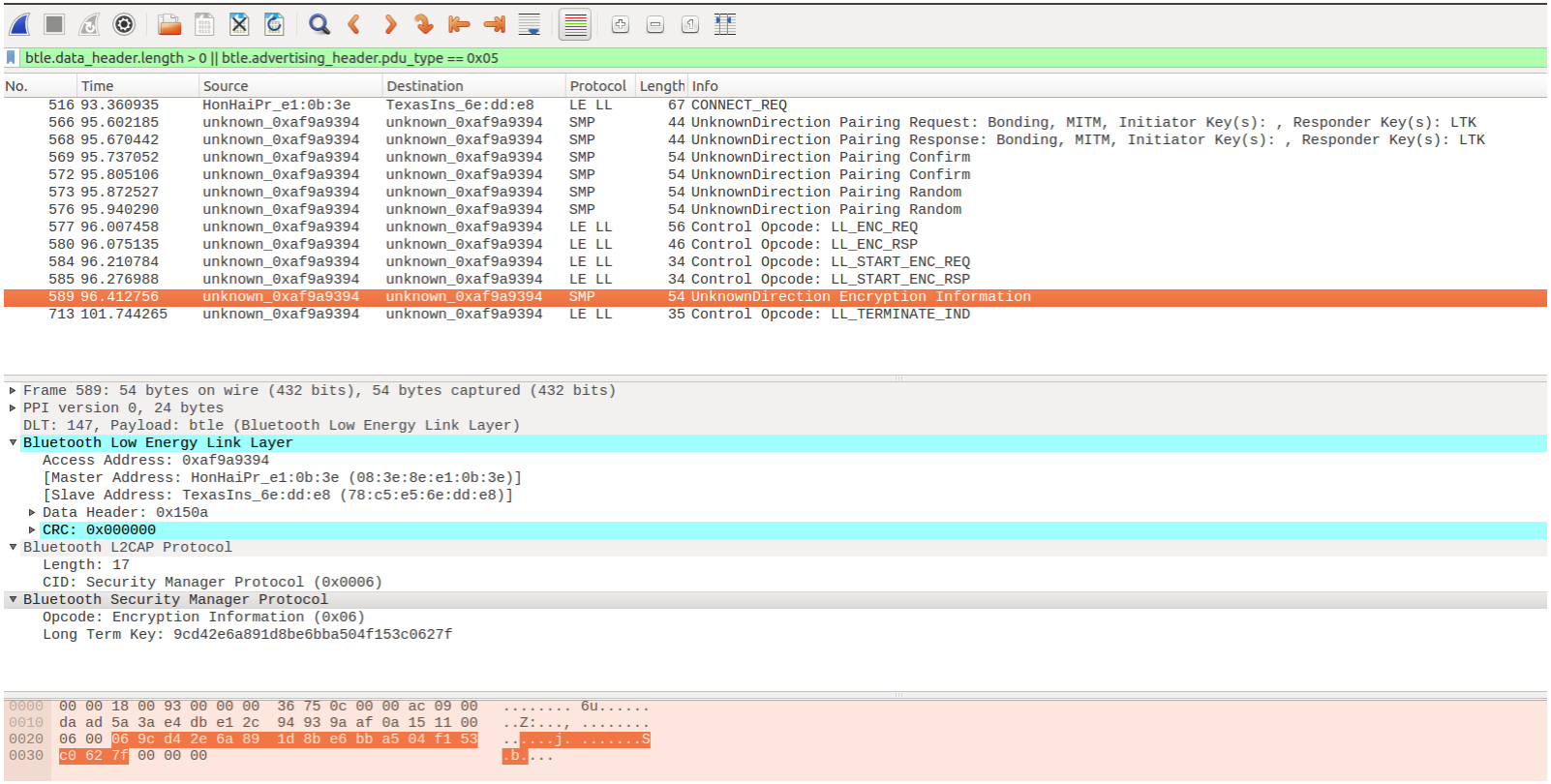
```
crackle -i pairing_and_ltk_exchange.pcap -o output.pcap
```

Expected output is in `out/`.


btle.data_header.length > 0 || btle.advertising_header.pdu_type == 0x05
No. Time Source Destination Protocol Length Info
516 93.360935 HonHaiPr_e1:0b:3e TexasIns_6e:dd:e8 LE LL 67 CONNECT_REQ
566 95.602185 unknown_0xaf9a9394 unknown_0xaf9a9394 SMP 44 UnknownDirection Pairing Request: Bonding, MITM, Initiator Key(s): , Responder Key(s): LTK
568 95.670442 unknown_0xaf9a9394 unknown_0xaf9a9394 SMP 44 UnknownDirection Pairing Response: Bonding, MITM, Initiator Key(s): , Responder Key(s): LTK
569 95.737052 unknown_0xaf9a9394 unknown_0xaf9a9394 SMP 54 UnknownDirection Pairing Confirm
572 95.805106 unknown_0xaf9a9394 unknown_0xaf9a9394 SMP 54 UnknownDirection Pairing Confirm
573 95.872527 unknown_0xaf9a9394 unknown_0xaf9a9394 SMP 54 UnknownDirection Pairing Random
576 95.940290 unknown_0xaf9a9394 unknown_0xaf9a9394 SMP 54 UnknownDirection Pairing Random
577 96.007458 unknown_0xaf9a9394 unknown_0xaf9a9394 LE LL 56 Control Opcode: LL_ENC_REQ
580 96.075135 unknown_0xaf9a9394 unknown_0xaf9a9394 LE LL 46 Control Opcode: LL_ENC_RSP
584 96.210784 unknown_0xaf9a9394 unknown_0xaf9a9394 LE LL 34 Control Opcode: LL_START_ENC_REQ
585 96.276988 unknown_0xaf9a9394 unknown_0xaf9a9394 LE LL 38 Control Opcode: Unknown
589 96.412756 unknown_0xaf9a9394 unknown_0xaf9a9394 LE LL 58 L2CAP Fragment
713 101.744265 unknown_0xaf9a9394 unknown_0xaf9a9394 LE LL 39 Control Opcode: Unknown
Frame 566: 44 bytes on wire (352 bits), 44 bytes captured (352 bits)
PPI version 0, 24 bytes
DLT: 147, Payload: btle (Bluetooth Low Energy Link Layer)
Bluetooth Low Energy Link Layer
Bluetooth L2CAP Protocol
Length: 7
CID: Security Manager Protocol (0x0006)
Bluetooth Security Manager Protocol
Opcode: Pairing Request (0x01)
IO Capability: No Input, No Output (0x03)
OOB Data Flags: OOB Auth. Data Not Present (0x00)
AuthReqBonding, MITM
Max Encryption Key Size: 16
Initiator Key Distribution
Responder Key DistributionLTK
0000 00 00 18 00 93 00 00 00 36 75 0c 00 00 80 09 00 .... 6u......
0010 f4 09 df 39 26 e1 f5 36 94 93 9a af 0e 0b 07 00 ...9&..6 ....
0020 06 00 01 03 00 05 10 00 01 f9 35 68 .. .. .5h


btle.data_header.length > 0 || btle.advertising_header.pdu_type == 0x05
No. Time Source Destination Protocol Length Info
516 93.360935 HonHaiPr_e1:0b:3e TexasIns_6e:dd:e8 LE LL 67 CONNECT_REQ
566 95.602185 unknown_0xaf9a9394 unknown_0xaf9a9394 SMP 44 UnknownDirection Pairing Request: Bonding, MITM, Initiator Key(s): , Responder Key(s): LTK
568 95.670442 unknown_0xaf9a9394 unknown_0xaf9a9394 SMP 44 UnknownDirection Pairing Response: Bonding, MITM, Initiator Key(s): , Responder Key(s): LTK
569 95.737052 unknown_0xaf9a9394 unknown_0xaf9a9394 SMP 54 UnknownDirection Pairing Confirm
572 95.805106 unknown_0xaf9a9394 unknown_0xaf9a9394 SMP 54 UnknownDirection Pairing Confirm
573 95.872527 unknown_0xaf9a9394 unknown_0xaf9a9394 SMP 54 UnknownDirection Pairing Random
576 95.940290 unknown_0xaf9a9394 unknown_0xaf9a9394 SMP 54 UnknownDirection Pairing Random
577 96.007458 unknown_0xaf9a9394 unknown_0xaf9a9394 LE LL 56 Control Opcode: LL_ENC_REQ
580 96.075135 unknown_0xaf9a9394 unknown_0xaf9a9394 LE LL 46 Control Opcode: LL_ENC_RSP
584 96.210784 unknown_0xaf9a9394 unknown_0xaf9a9394 LE LL 34 Control Opcode: LL_START_ENC_REQ
585 96.276988 unknown_0xaf9a9394 unknown_0xaf9a9394 LE LL 38 Control Opcode: Unknown
589 96.412756 unknown_0xaf9a9394 unknown_0xaf9a9394 LE LL 58 L2CAP Fragment
713 101.744265 unknown_0xaf9a9394 unknown_0xaf9a9394 LE LL 39 Control Opcode: Unknown
Frame 568: 44 bytes on wire (352 bits), 44 bytes captured (352 bits)
PPI version 0, 24 bytes
DLT: 147, Payload: btle (Bluetooth Low Energy Link Layer)
Bluetooth Low Energy Link Layer
Bluetooth L2CAP Protocol
Length: 7
CID: Security Manager Protocol (0x0006)
Bluetooth Security Manager Protocol
Opcode: Pairing Response (0x02)
IO Capability: Display Only (0x00)
OOB Data Flags: OOB Auth. Data Not Present (0x00)
AuthReqBonding, MITM
Max Encryption Key Size: 16
Initiator Key Distribution
Responder Key DistributionLTK
0000 00 00 18 00 93 00 00 00 36 75 0c 00 00 90 09 00 .... 6u......
0010 60 5f e9 39 ed df f2 2b 94 93 9a af 06 0b 07 00 `_.9...+ ....
0020 06 00 02 00 00 05 10 00 01 a2 fc 02 ....


csc@ubuntu: ~/crackle-master/tests/01_crack
02_ltk_decrypt 04_le_secure_connections run_tests.pl
csc@ubuntu:~/crackle-master/tests$ cd 01_crack/
csc@ubuntu:~/crackle-master/tests/01_crack$ ls
out pairing_and_ltk_exchange.pcap README.md
csc@ubuntu:~/crackle-master/tests/01_crack$ crackle -i pairing_and_ltk_exchange.
pcap -o output.pcap
Found 1 connection
Analyzing connection 0:
08:3e:8e:e1:0b:3e (public) -> 78:c5:e5:6e:dd:e8 (public)
Found 3 encrypted packets
Cracking with strategy 0, 20 bits of entropy
!!!
TK found: 000000
ding ding ding, using a TK of 0! Just Cracks(tm)
!!!
Decrypted 3 packets
LTK found: 7f62c053f104a5bbe68b1d896a2ed49c
Decrypted 3 packets, dumping to PCAP
Done, processed 713 total packets, decrypted 3
csc@ubuntu:~/crackle-master/tests/01_crack$


btle.data_header.length > 0 || btle.advertising_header.pdu_type == 0x05
No. Time Source Destination Protocol Length Info
516 93.360935 HonHaiPr_e1:0b:3e TexasIns_6e:dd:e8 LE LL 67 CONNECT_REQ
566 95.602185 unknown_0xaf9a9394 unknown_0xaf9a9394 SMP 44 UnknownDirection Pairing Request: Bonding, MITM, Initiator Key(s): , Responder Key(s): LTK
568 95.670442 unknown_0xaf9a9394 unknown_0xaf9a9394 SMP 44 UnknownDirection Pairing Response: Bonding, MITM, Initiator Key(s): , Responder Key(s): LTK
569 95.737052 unknown_0xaf9a9394 unknown_0xaf9a9394 SMP 54 UnknownDirection Pairing Confirm
572 95.805106 unknown_0xaf9a9394 unknown_0xaf9a9394 SMP 54 UnknownDirection Pairing Confirm
573 95.872527 unknown_0xaf9a9394 unknown_0xaf9a9394 SMP 54 UnknownDirection Pairing Random
576 95.940290 unknown_0xaf9a9394 unknown_0xaf9a9394 SMP 54 UnknownDirection Pairing Random
577 96.007458 unknown_0xaf9a9394 unknown_0xaf9a9394 LE LL 56 Control Opcode: LL_ENC_REQ
580 96.075135 unknown_0xaf9a9394 unknown_0xaf9a9394 LE LL 46 Control Opcode: LL_ENC_RSP
584 96.210784 unknown_0xaf9a9394 unknown_0xaf9a9394 LE LL 34 Control Opcode: LL_START_ENC_REQ
585 96.276988 unknown_0xaf9a9394 unknown_0xaf9a9394 LE LL 34 Control Opcode: LL_START_ENC_RSP
589 96.412756 unknown_0xaf9a9394 unknown_0xaf9a9394 SMP 54 UnknownDirection Encryption Information
713 101.744265 unknown_0xaf9a9394 unknown_0xaf9a9394 LE LL 35 Control Opcode: LL_TERMINATE_IND
Frame 589: 54 bytes on wire (432 bits), 54 bytes captured (432 bits)
PPI version 0, 24 bytes
DLT: 147, Payload: btle (Bluetooth Low Energy Link Layer)
Bluetooth Low Energy Link Layer
Access Address: 0xaf9a9394
[Master Address: HonHaiPr_e1:0b:3e (08:3e:8e:e1:0b:3e)]
[Slave Address: TexasIns_6e:dd:e8 (78:c5:e5:6e:dd:e8)]
Data Header: 0x150a
CRC: 0x000000
Bluetooth L2CAP Protocol
Length: 17
CID: Security Manager Protocol (0x0006)
Bluetooth Security Manager Protocol
Opcode: Encryption Information (0x06)
Long Term Key: 9cd42e6a891d8be6bba504f153c0627f
0000 00 00 18 00 93 00 00 00 36 75 0c 00 00 ac 09 00 .... 6u......
0010 da ad 5a 3a e4 db e1 2c 94 93 9a af 0a 15 11 00 ..Z:..., ...
0020 06 00 06 9c d4 2e 6a 89 1d 8b e6 bb a5 04 f1 53 ......j. ....S
0030 c0 62 7f 00 00 00 .b....

mikeryan / crackle

Watch 42 | Star 298 | Fork 53

Code | Issues 3 | Pull requests 1 | Projects 0 | Wiki | Insights

Branch: master ▾ crackle / tests / 02_ltk_decrypt / Create new file | Upload files | Find file | History

mikeryan bugfix: decrypt packets of length 5 or greater ... Latest commit 10c6d65 on 15 Jan

..		
out	bugfix: decrypt packets of length 5 or greater	7 months ago
README.md	tests: add test harness and basic test cases	10 months ago
args.txt	tests: add test harness and basic test cases	10 months ago
known_ltk.pcap	tests: add test harness and basic test cases	10 months ago

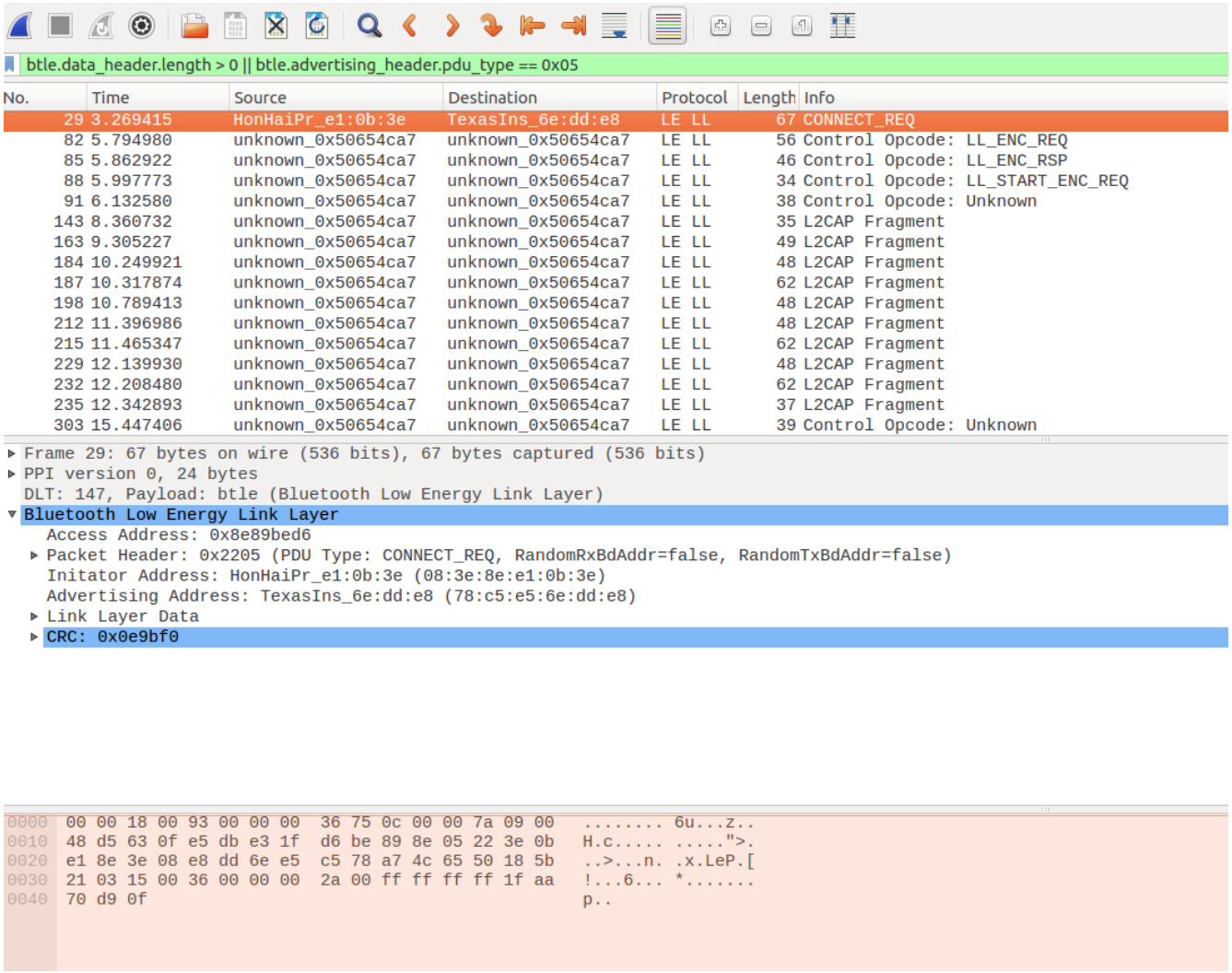
README.md

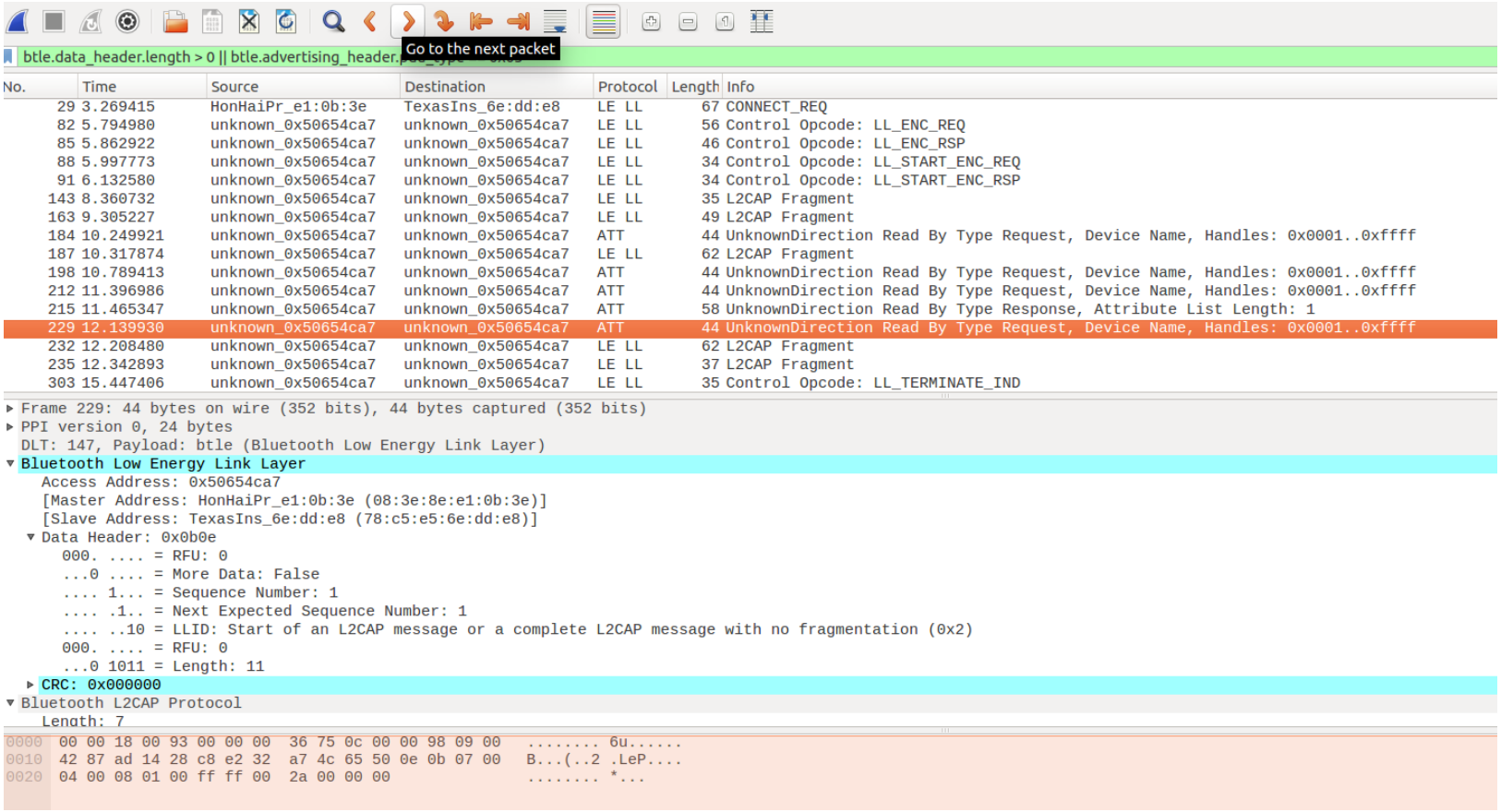
This test validates crackle's LTK decrypt functionality. The packets in this sample PCAP file are encrypted with the LTK from test 01. By running crackle in LTK decrypt mode with this LTK, crackle will successfully decrypt most of the encrypted packets.

To run this test manually:

```
crackle -i known_ltk.pcap -o output.pcap -l 7f62c053f104a5bbe68b1d896a2ed49c
```

Expected output is in `out/`.


btle.data_header.length > 0 || btle.advertising_header.pdu_type == 0x05
No. Time Source Destination Protocol Length Info
29 3.269415 HonHaiPr_e1:0b:3e TexasIns_6e:dd:e8 LE LL 67 CONNECT_REQ
82 5.794980 unknown_0x50654ca7 unknown_0x50654ca7 LE LL 56 Control Opcode: LL_ENC_REQ
85 5.862922 unknown_0x50654ca7 unknown_0x50654ca7 LE LL 46 Control Opcode: LL_ENC_RSP
88 5.997773 unknown_0x50654ca7 unknown_0x50654ca7 LE LL 34 Control Opcode: LL_START_ENC_REQ
91 6.132580 unknown_0x50654ca7 unknown_0x50654ca7 LE LL 38 Control Opcode: Unknown
143 8.360732 unknown_0x50654ca7 unknown_0x50654ca7 LE LL 35 L2CAP Fragment
163 9.305227 unknown_0x50654ca7 unknown_0x50654ca7 LE LL 49 L2CAP Fragment
184 10.249921 unknown_0x50654ca7 unknown_0x50654ca7 LE LL 48 L2CAP Fragment
187 10.317874 unknown_0x50654ca7 unknown_0x50654ca7 LE LL 62 L2CAP Fragment
198 10.789413 unknown_0x50654ca7 unknown_0x50654ca7 LE LL 48 L2CAP Fragment
212 11.396986 unknown_0x50654ca7 unknown_0x50654ca7 LE LL 48 L2CAP Fragment
215 11.465347 unknown_0x50654ca7 unknown_0x50654ca7 LE LL 62 L2CAP Fragment
229 12.139930 unknown_0x50654ca7 unknown_0x50654ca7 LE LL 48 L2CAP Fragment
232 12.208480 unknown_0x50654ca7 unknown_0x50654ca7 LE LL 62 L2CAP Fragment
235 12.342893 unknown_0x50654ca7 unknown_0x50654ca7 LE LL 37 L2CAP Fragment
303 15.447406 unknown_0x50654ca7 unknown_0x50654ca7 LE LL 39 Control Opcode: Unknown
Frame 29: 67 bytes on wire (536 bits), 67 bytes captured (536 bits)
PPI version 0, 24 bytes
DLT: 147, Payload: btle (Bluetooth Low Energy Link Layer)
Bluetooth Low Energy Link Layer
Access Address: 0x8e89bed6
Packet Header: 0x2205 (PDU Type: CONNECT_REQ, RandomRxBdAddr=false, RandomTxBdAddr=false)
Initator Address: HonHaiPr_e1:0b:3e (08:3e:8e:e1:0b:3e)
Advertising Address: TexasIns_6e:dd:e8 (78:c5:e5:6e:dd:e8)
Link Layer Data
CRC: 0x0e9bf0
0000 00 00 18 00 93 00 00 00 36 75 0c 00 00 7a 09 00 .... 6u...z..
0010 48 d5 63 0f e5 db e3 1f d6 be 89 8e 05 22 3e 0b H.c..... .....">.
0020 e1 8e 3e 08 e8 dd 6e e5 c5 78 a7 4c 65 50 18 5b ..>...n. .x.LeP.[
0030 21 03 15 00 36 00 00 00 2a 00 ff ff ff ff 1f aa !...6... *.......
0040 70 d9 0f p..


Go to the next packet
btle.data_header.length > 0 || btle.advertising_header.
No. Time Source Destination Protocol Length Info
29 3.269415 HonHaiPr_e1:0b:3e TexasIns_6e:dd:e8 LE LL 67 CONNECT_REQ
82 5.794980 unknown_0x50654ca7 unknown_0x50654ca7 LE LL 56 Control Opcode: LL_ENC_REQ
85 5.862922 unknown_0x50654ca7 unknown_0x50654ca7 LE LL 46 Control Opcode: LL_ENC_RSP
88 5.997773 unknown_0x50654ca7 unknown_0x50654ca7 LE LL 34 Control Opcode: LL_START_ENC_REQ
91 6.132580 unknown_0x50654ca7 unknown_0x50654ca7 LE LL 34 Control Opcode: LL_START_ENC_RSP
143 8.360732 unknown_0x50654ca7 unknown_0x50654ca7 LE LL 35 L2CAP Fragment
163 9.305227 unknown_0x50654ca7 unknown_0x50654ca7 LE LL 49 L2CAP Fragment
184 10.249921 unknown_0x50654ca7 unknown_0x50654ca7 ATT 44 UnknownDirection Read By Type Request, Device Name, Handles: 0x0001..0xffff
187 10.317874 unknown_0x50654ca7 unknown_0x50654ca7 LE LL 62 L2CAP Fragment
198 10.789413 unknown_0x50654ca7 unknown_0x50654ca7 ATT 44 UnknownDirection Read By Type Request, Device Name, Handles: 0x0001..0xffff
212 11.396986 unknown_0x50654ca7 unknown_0x50654ca7 ATT 44 UnknownDirection Read By Type Request, Device Name, Handles: 0x0001..0xffff
215 11.465347 unknown_0x50654ca7 unknown_0x50654ca7 ATT 58 UnknownDirection Read By Type Response, Attribute List Length: 1
229 12.139930 unknown_0x50654ca7 unknown_0x50654ca7 ATT 44 UnknownDirection Read By Type Request, Device Name, Handles: 0x0001..0xffff
232 12.208480 unknown_0x50654ca7 unknown_0x50654ca7 LE LL 62 L2CAP Fragment
235 12.342893 unknown_0x50654ca7 unknown_0x50654ca7 LE LL 37 L2CAP Fragment
303 15.447406 unknown_0x50654ca7 unknown_0x50654ca7 LE LL 35 Control Opcode: LL_TERMINATE_IND
Frame 229: 44 bytes on wire (352 bits), 44 bytes captured (352 bits)
PPI version 0, 24 bytes
DLT: 147, Payload: btle (Bluetooth Low Energy Link Layer)
Bluetooth Low Energy Link Layer
Access Address: 0x50654ca7
[Master Address: HonHaiPr_e1:0b:3e (08:3e:8e:e1:0b:3e)]
[Slave Address: TexasIns_6e:dd:e8 (78:c5:e5:6e:dd:e8)]
Data Header: 0x0b0e
000. .... = RFU: 0
...0 .... = More Data: False
.... 1... = Sequence Number: 1
.... .1.. = Next Expected Sequence Number: 1
.... ..10 = LLID: Start of an L2CAP message or a complete L2CAP message with no fragmentation (0x2)
000. .... = RFU: 0
...0 1011 = Length: 11
CRC: 0x000000
Bluetooth L2CAP Protocol
Length: 7
0000 00 00 18 00 93 00 00 00 36 75 0c 00 00 98 09 00 .... 6u......
0010 42 87 ad 14 28 c8 e2 32 a7 4c 65 50 0e 0b 07 00 B...(..2 .LeP....
0020 04 00 08 01 00 ff ff 00 2a 00 00 00 .... *...

mikeryan / crackle

Watch 42 | Star 298 | Fork 53

Code | Issues 3 | Pull requests 1 | Projects 0 | Wiki | Insights

Branch: master ▾ crackle / tests / 04_le_secure_connections /

Create new file | Upload files | Find file | History

mikeryan correctly indicate random BD ADDRs ... Latest commit a208476 on 14 Mar

..

out	correctly indicate random BD ADDRs	6 months ago
README.md	tests: add test harness and basic test cases	10 months ago
le_secure_connections.pcapng	tests: add test harness and basic test cases	10 months ago

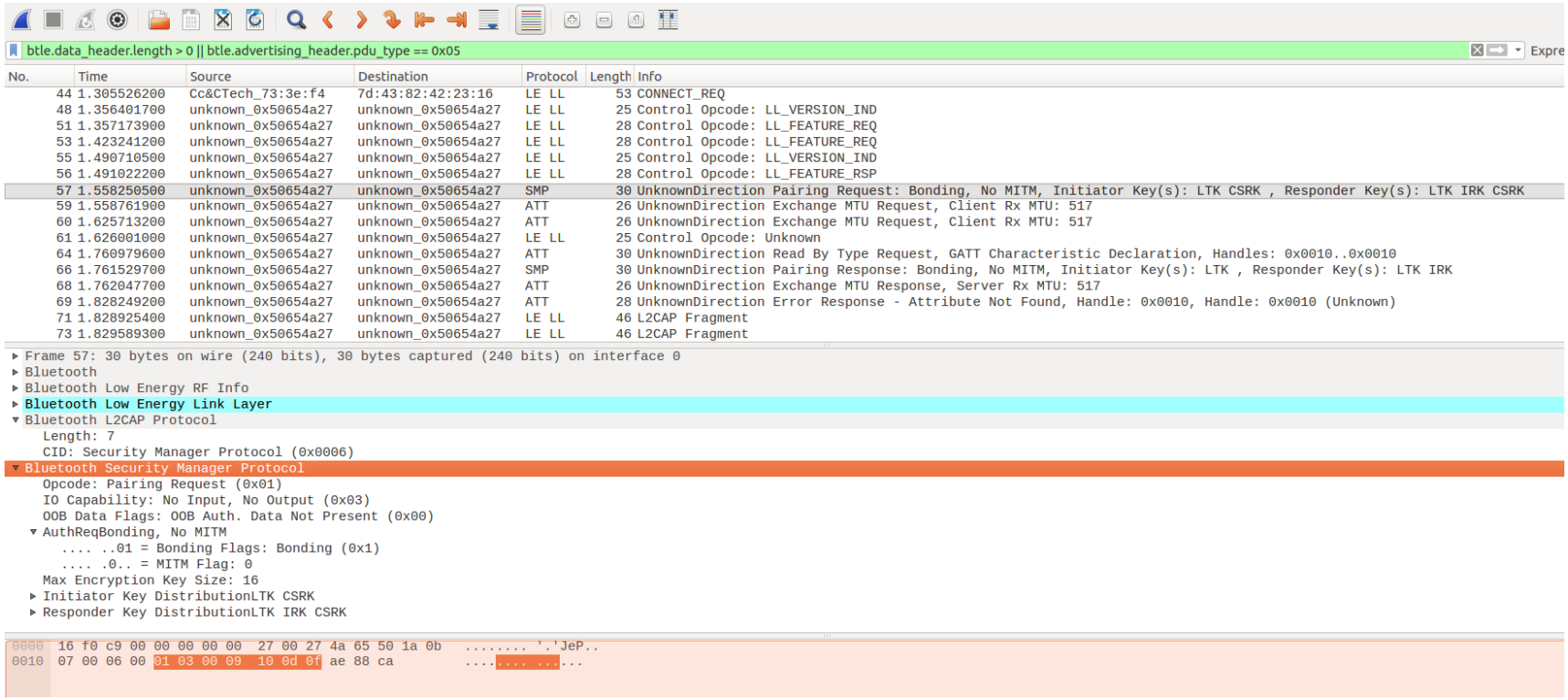
README.md

This test validates that crackle correctly detects when LE Secure Connections are in use. Upon discovering an LE SC pairing packet, crackle will immediately throw an error and fail to crack the pairing.

To run this test manually:

```
crackle -i le_secure_connections.pcapng
```

Expected output is in `out/`.


btle.data_header.length > 0 || btle.advertising_header.pdu_type == 0x05
No. Time Source Destination Protocol Length Info
44 1.305526200 Cc&CTech_73:3e:f4 7d:43:82:42:23:16 LE LL 53 CONNECT_REQ
48 1.356401700 unknown_0x50654a27 unknown_0x50654a27 LE LL 25 Control Opcode: LL_VERSION_IND
51 1.357173900 unknown_0x50654a27 unknown_0x50654a27 LE LL 28 Control Opcode: LL_FEATURE_REQ
53 1.423241200 unknown_0x50654a27 unknown_0x50654a27 LE LL 28 Control Opcode: LL_FEATURE_REQ
55 1.490710500 unknown_0x50654a27 unknown_0x50654a27 LE LL 25 Control Opcode: LL_VERSION_IND
56 1.491022200 unknown_0x50654a27 unknown_0x50654a27 LE LL 28 Control Opcode: LL_FEATURE_RSP
57 1.558250500 unknown_0x50654a27 unknown_0x50654a27 SMP 30 UnknownDirection Pairing Request: Bonding, No MITM, Initiator Key(s): LTK CSRK , Responder Key(s): LTK IRK CSRK
59 1.558761900 unknown_0x50654a27 unknown_0x50654a27 ATT 26 UnknownDirection Exchange MTU Request, Client Rx MTU: 517
60 1.625713200 unknown_0x50654a27 unknown_0x50654a27 ATT 26 UnknownDirection Exchange MTU Request, Client Rx MTU: 517
61 1.626001000 unknown_0x50654a27 unknown_0x50654a27 LE LL 25 Control Opcode: Unknown
64 1.760979600 unknown_0x50654a27 unknown_0x50654a27 ATT 30 UnknownDirection Read By Type Request, GATT Characteristic Declaration, Handles: 0x0010..0x0010
66 1.761529700 unknown_0x50654a27 unknown_0x50654a27 SMP 30 UnknownDirection Pairing Response: Bonding, No MITM, Initiator Key(s): LTK , Responder Key(s): LTK IRK
68 1.762047700 unknown_0x50654a27 unknown_0x50654a27 ATT 26 UnknownDirection Exchange MTU Response, Server Rx MTU: 517
69 1.828249200 unknown_0x50654a27 unknown_0x50654a27 ATT 28 UnknownDirection Error Response - Attribute Not Found, Handle: 0x0010, Handle: 0x0010 (Unknown)
71 1.828925400 unknown_0x50654a27 unknown_0x50654a27 LE LL 46 L2CAP Fragment
73 1.829589300 unknown_0x50654a27 unknown_0x50654a27 LE LL 46 L2CAP Fragment
Frame 57: 30 bytes on wire (240 bits), 30 bytes captured (240 bits) on interface 0
Bluetooth
Bluetooth Low Energy RF Info
Bluetooth Low Energy Link Layer
Bluetooth L2CAP Protocol
Length: 7
CID: Security Manager Protocol (0x0006)
Bluetooth Security Manager Protocol
Opcode: Pairing Request (0x01)
IO Capability: No Input, No Output (0x03)
OOB Data Flags: OOB Auth. Data Not Present (0x00)
AuthReqBonding, No MITM
.... ..01 = Bonding Flags: Bonding (0x1)
.... .0.. = MITM Flag: 0
Max Encryption Key Size: 16
Initiator Key DistributionLTK CSRK
Responder Key DistributionLTK IRK CSRK
0000 16 f0 c9 00 00 00 00 00 27 00 27 4a 65 50 1a 0b
0010 07 00 06 00 01 03 00 09 10 0d 0f ae 88 ca

練習四：判斷連線種類

- 取得 Wireshark 的封包
- 讀取封包看到底是哪一種連線方式

4 結論與未來方向

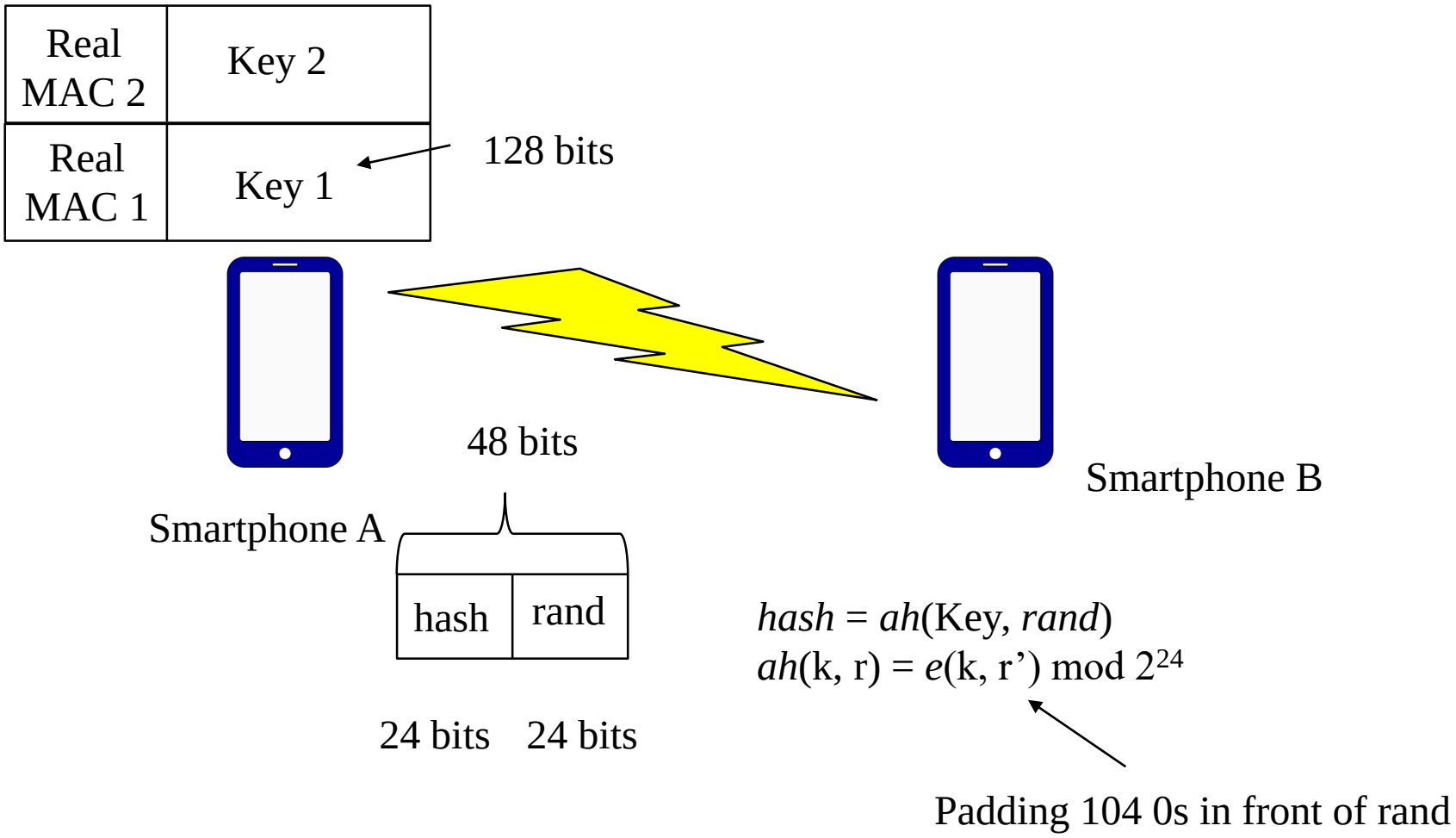
結論

- 因為智慧型手機應用的普及，很多物聯網裝置會使用BLE 協定
- 藍牙基本運作模式包括：
  - 廣播/ 觀察者模式
  - 周邊/ 中央模式
- BLE 協定若未考慮到安全性，則有可能形成弱點
  - 當然可以採用最高等級的安全協定，但是不見得所有的裝置都支援
  - 所以目前很多裝置都不一定使用標準安全協定
- 透過實驗我們可以體會一些常見的安全威脅

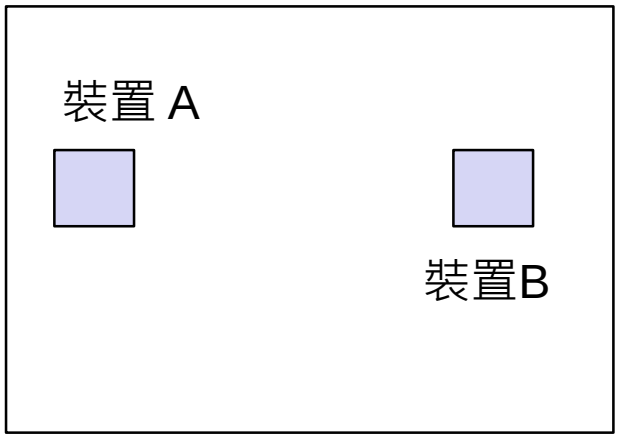
下一步

- 了解 BLE 的其他運作機制
  - 例如隱私位置
- 了解到分析 BLE 裝置之方法
- 補強實體安全分析做法

BLE 中的可還原亂數位置



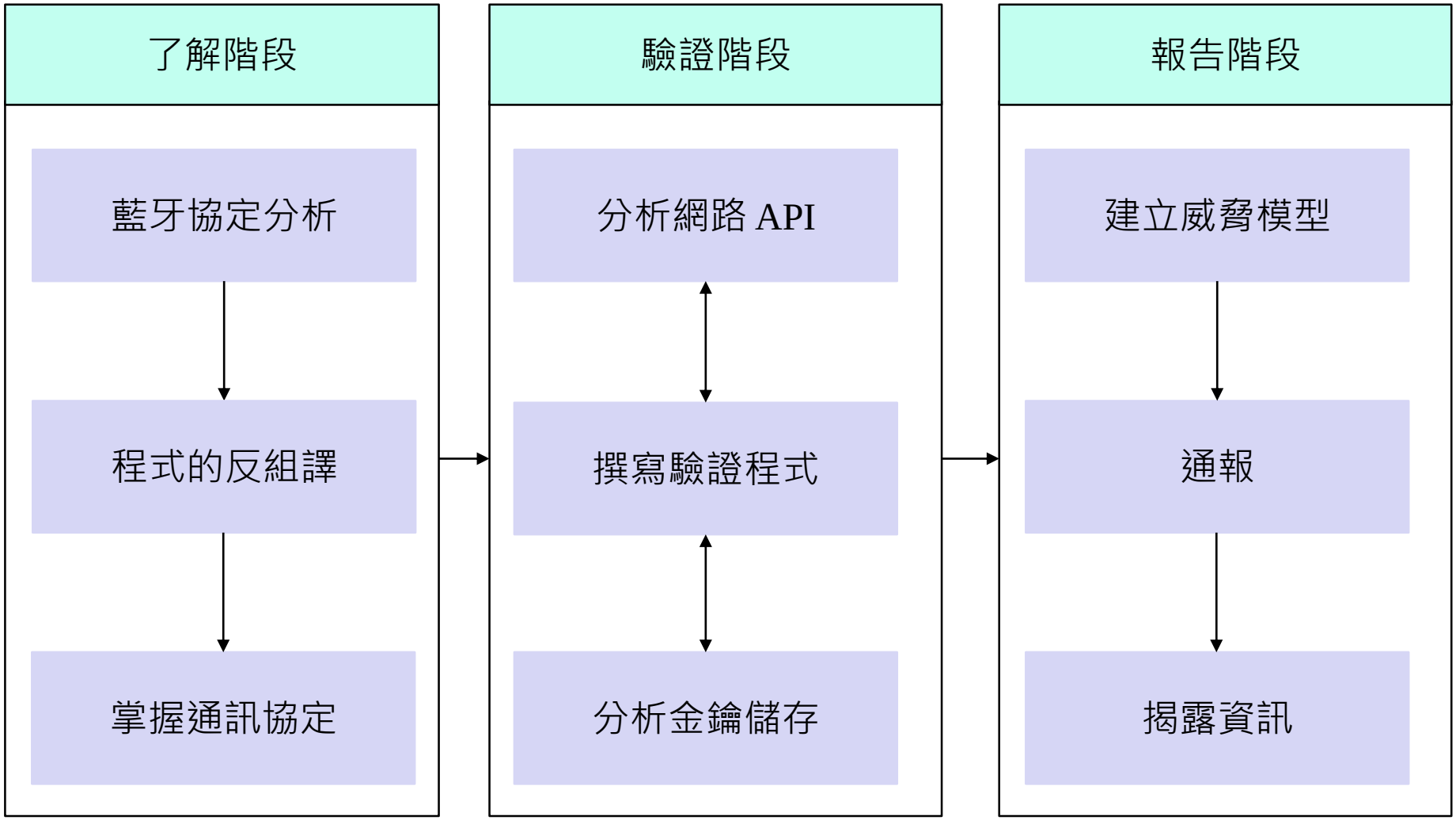
減少雜訊，集中關注待測標的



解析傳播模式

- 誰發起廣播/掃描？
- 是否有連線？
- 連線方式與安全性

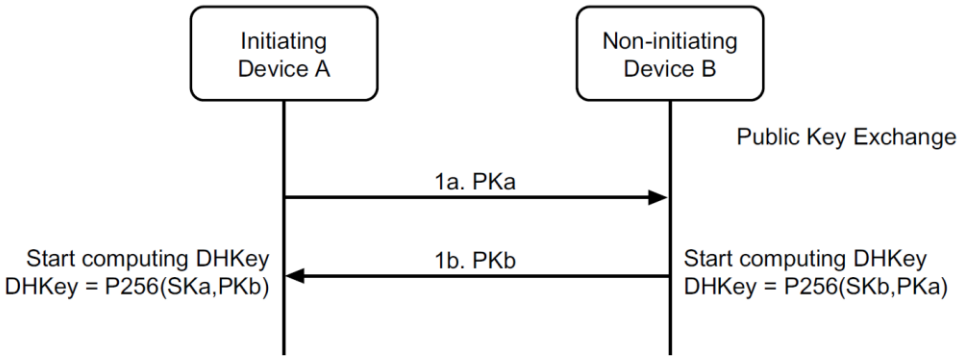
尋找弱點與確認


了解階段
藍牙協定分析
程式的反組譯
掌握通訊協定
驗證階段
分析網路 API
撰寫驗證程式
分析金鑰儲存
報告階段
建立威脅模型
通報
揭露資訊

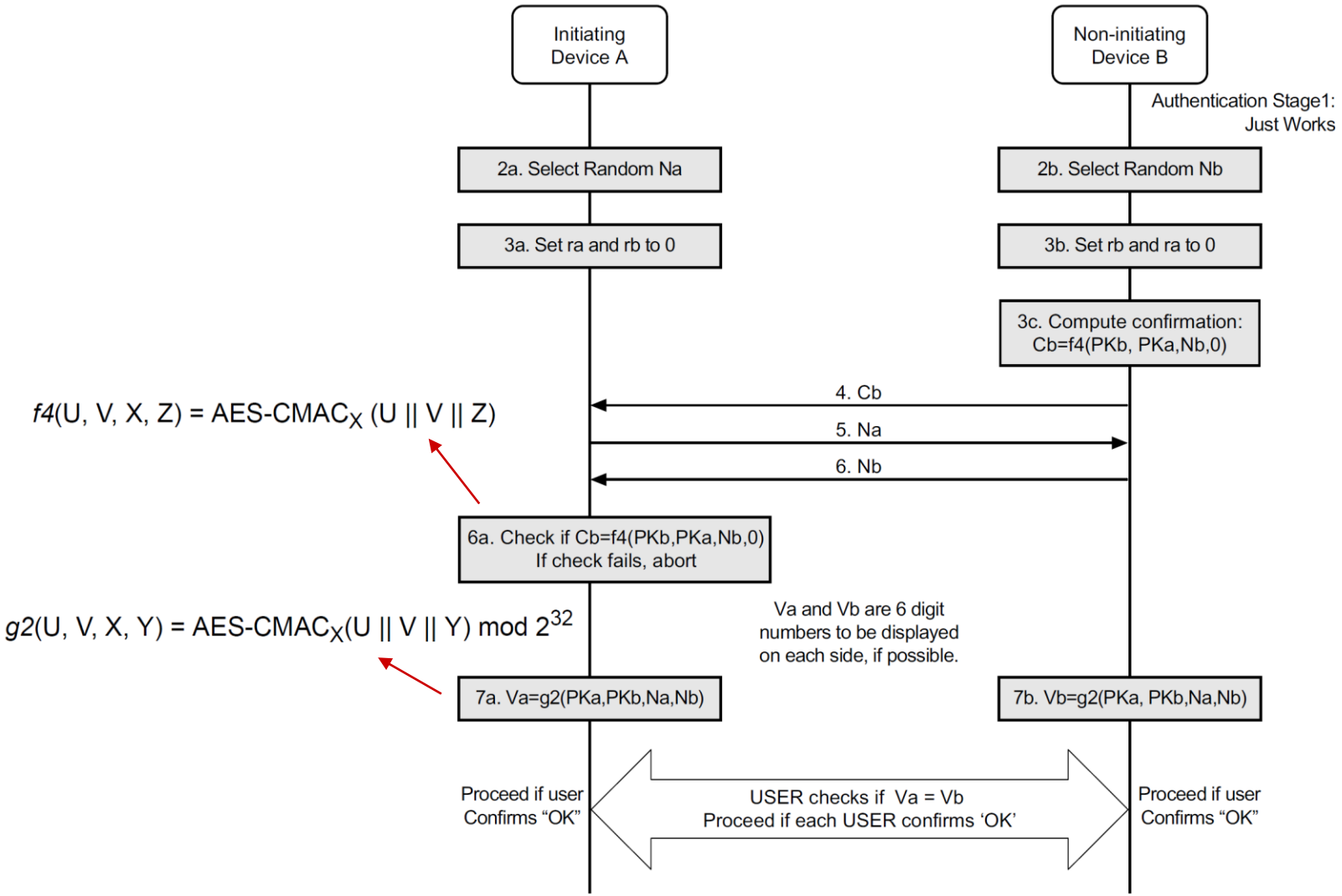
Thank you

走 LE Secure Connection 時的 Phase 2

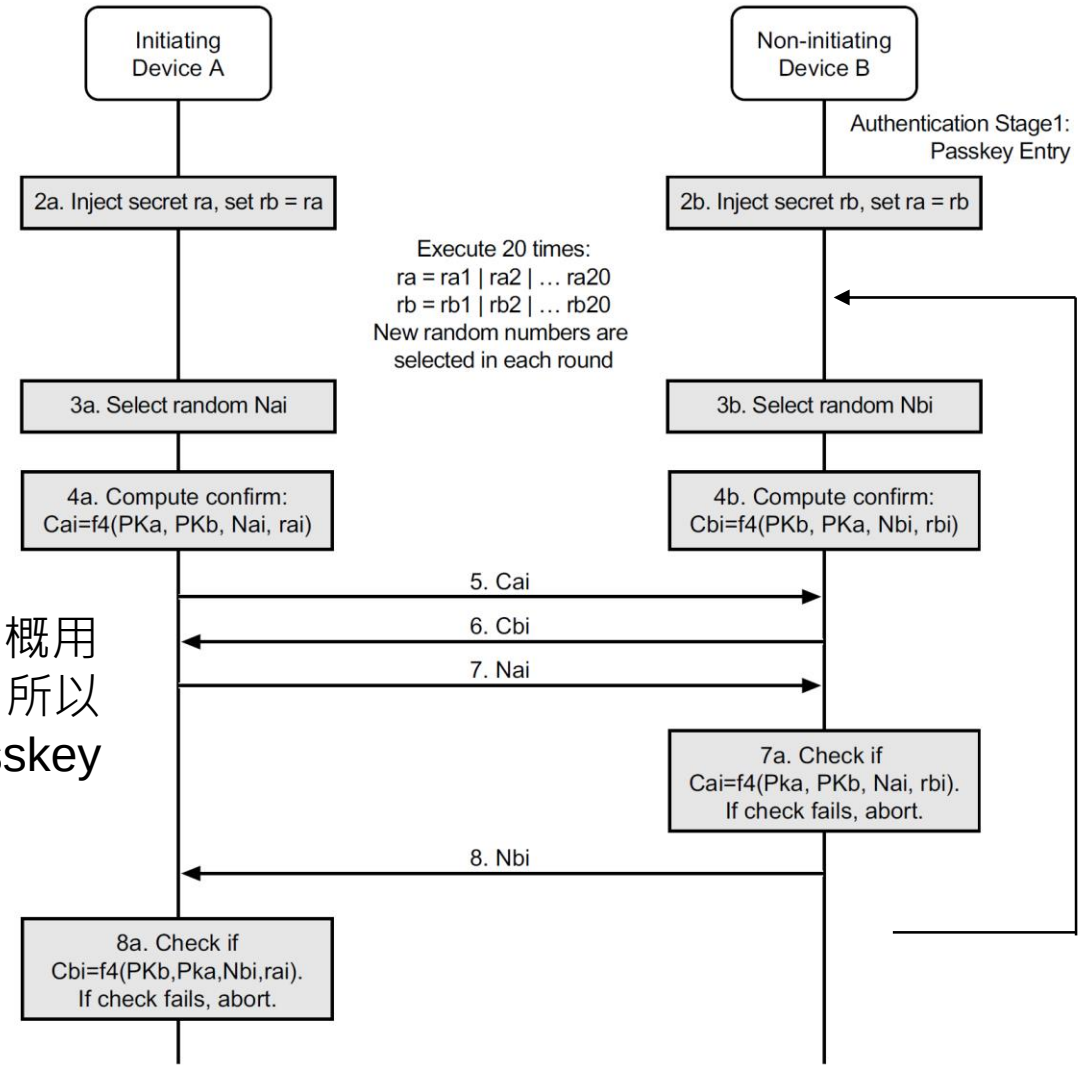
- 基本上會用 ECCDH 的方式先產生共通的金鑰



在 LE Secure Connection 底下 Just Works 與 Numeric Comparison 的方法



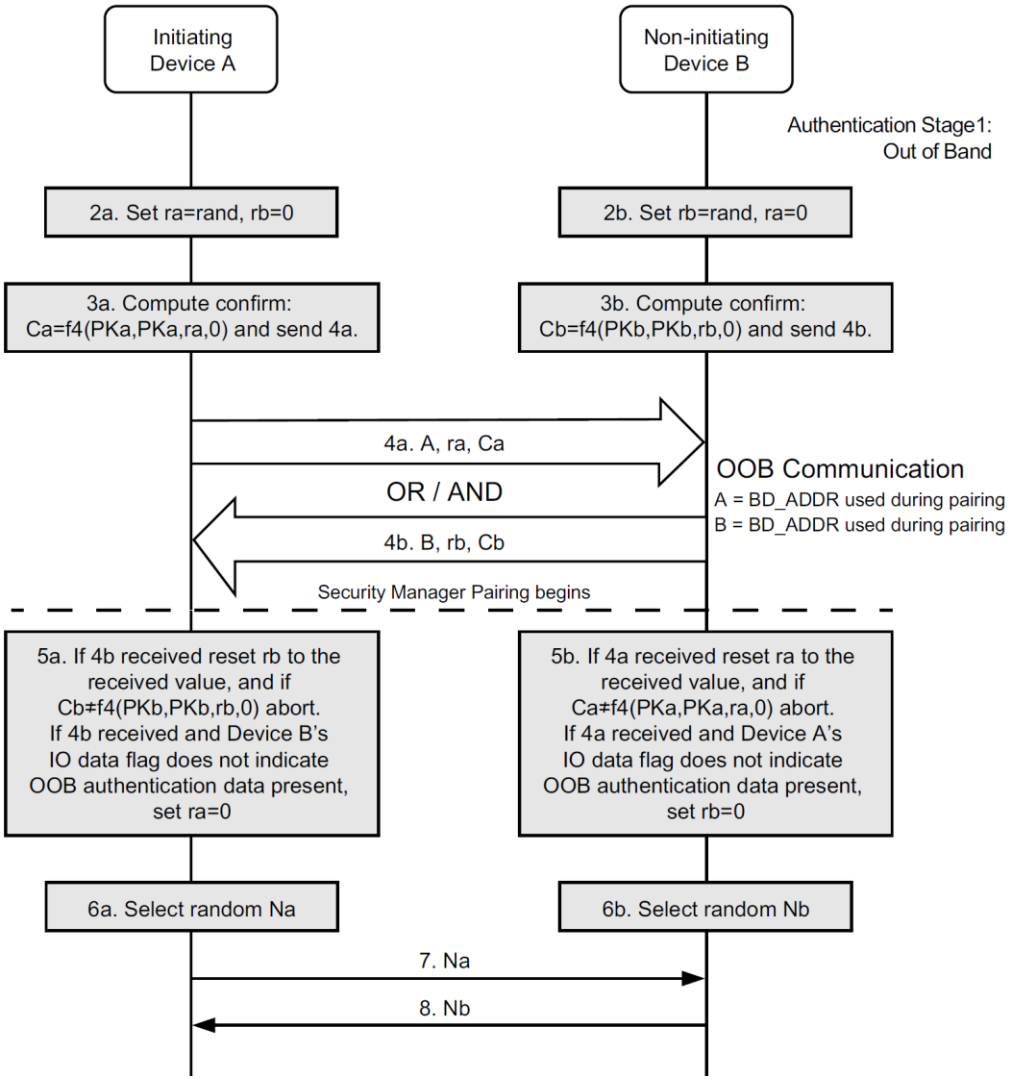
LE Secure Connection 的 Passkey 方法比較麻煩



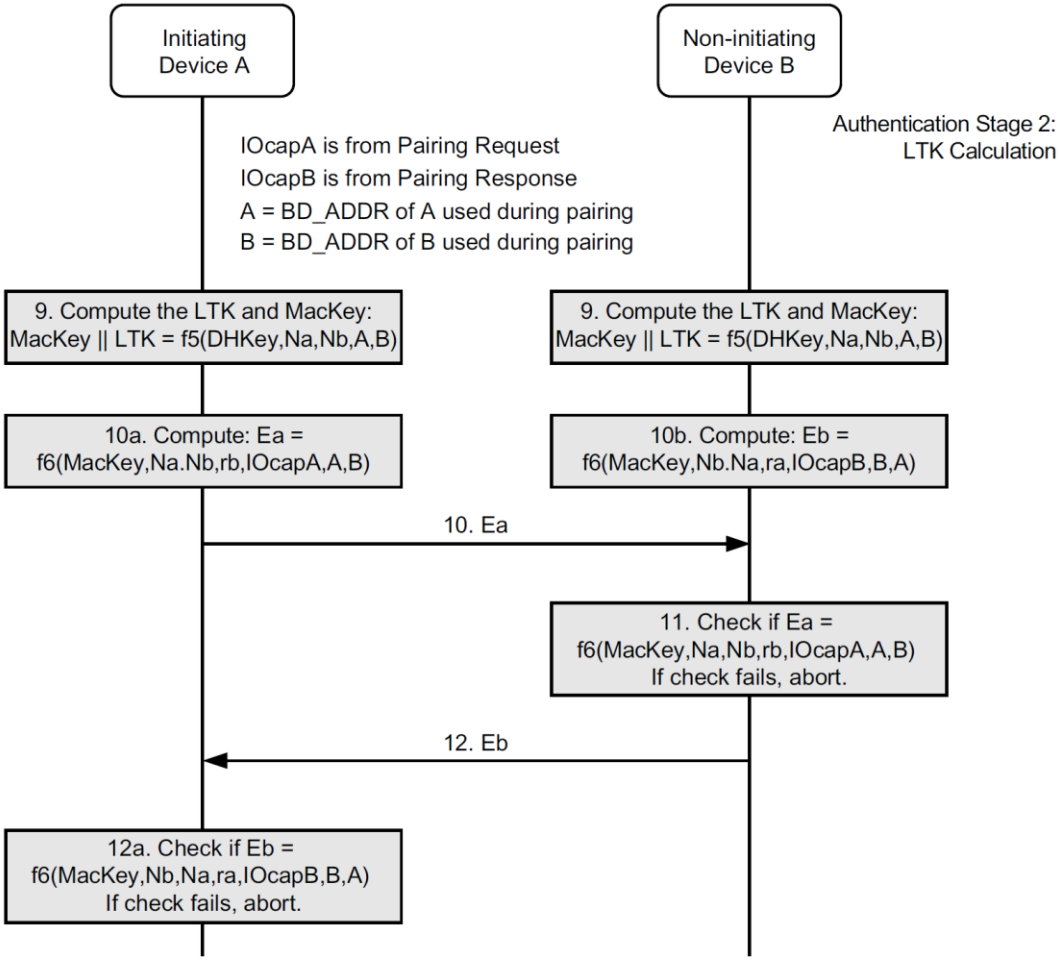
因為 999999 大概用 20bits 可表示，所以這邊就是把 passkey 每個 bit 算一次

這邊會循環 20 次

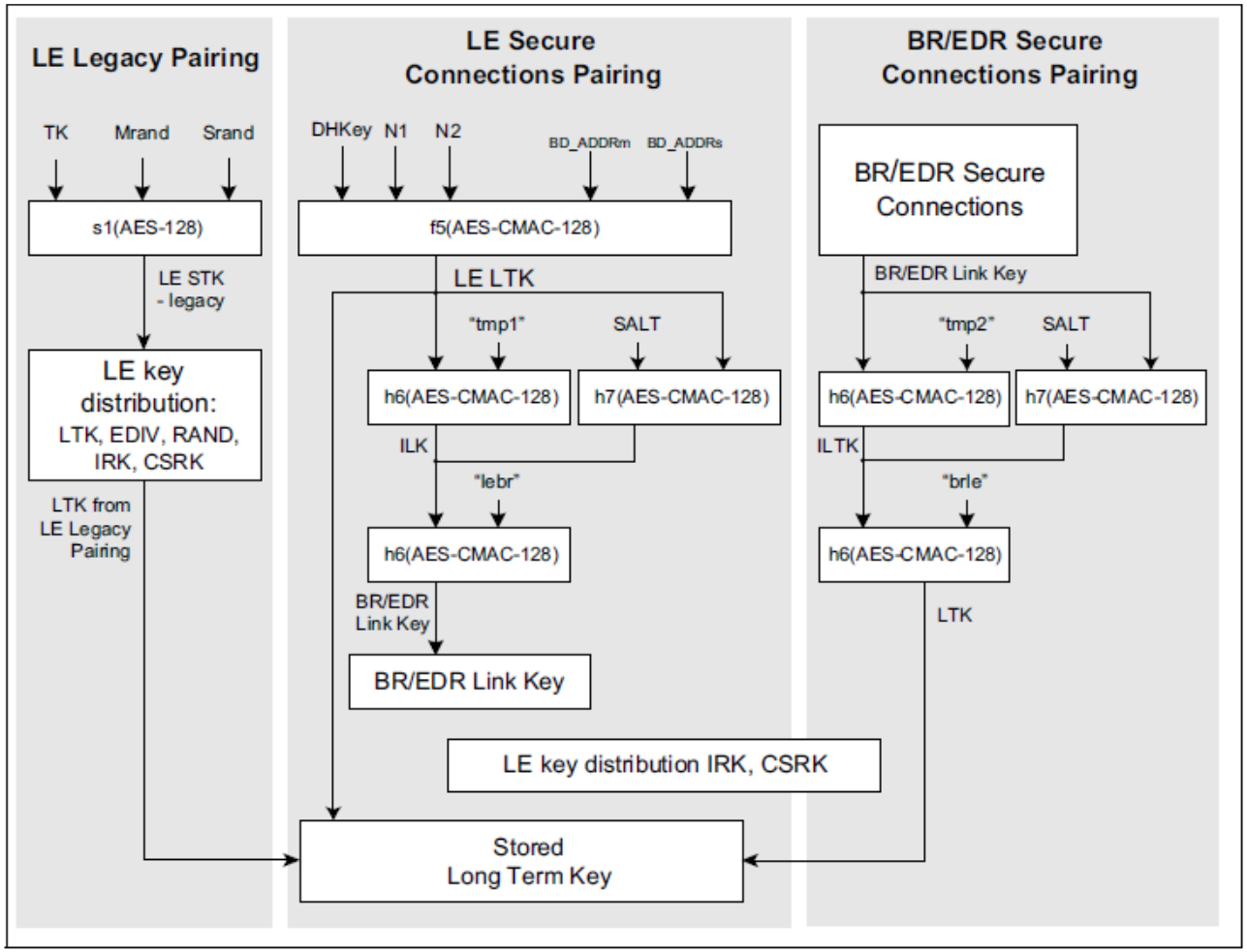
OOB 和 Just Works 相似，就是那個亂數怎麼拿到對面去



然後接下來產生之後用的 LTK



彙整的概念


LE Legacy Pairing
TK
Mrand
Srand
s1(AES-128)
LE STK
- legacy
LE key
distribution:
LTK, EDIV, RAND,
IRK, CSRK
LTK from
LE Legacy
Pairing
LE Secure
Connections Pairing
DHKey
N1
N2
BD_ADDRm
BD_ADDRs
f5(AES-CMAC-128)
LE LTK
"tmp1"
SALT
h6(AES-CMAC-128)
h7(AES-CMAC-128)
ILK
"lebr"
h6(AES-CMAC-128)
BR/EDR
Link Key
BR/EDR Link Key
BR/EDR Secure
Connections Pairing
BR/EDR Secure
Connections
BR/EDR Link Key
"tmp2"
SALT
h6(AES-CMAC-128)
h7(AES-CMAC-128)
ILTK
"brle"
h6(AES-CMAC-128)
LTK
LE key distribution IRK, CSRK
Stored
Long Term Key