Extracting the painful (blue)tooth

Matteo Beccaro

- **Security Consultant** at **Secure Network**
- **Technical Research Leader** at **Opposing Force**, Physical Security division of Secure Network
- **@_bughardy_**

Matteo Collura

- **Student at Politecnico di Torino (www.polito.it)**
- **Electronic Engineer**
- **Researcher in several fields**
- **@eagle1753**

Who we are…

{Matteo Beccaro

- **Security Consultant** at **Secure Network**
- **Technical Research Leader** at **Opposing Force**, Physical Security division of Secure Network
- **@_bughardy_**

OPPOSING FORCE
challenging your security

secure network
Your Protection, Our Mission

Who are we…

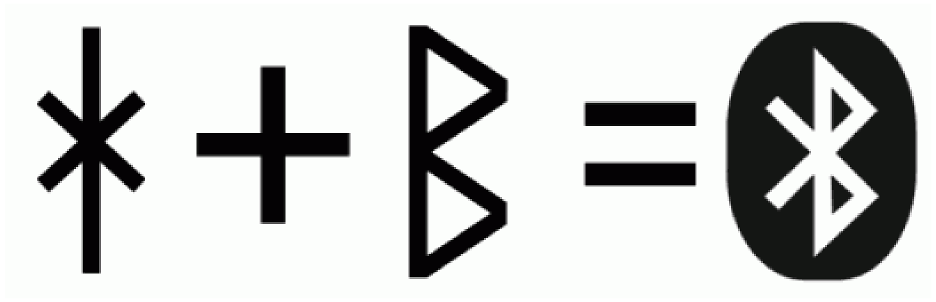
Matteo Collura

- Student at Politecnico di Torino
- Electronic Engineer
- Researcher in different fields concerning security (NFC, bluetooth)
- Now focusing on social skills (NLP, social engineering..)
- @eagle1753

Index

What the hell is Bluetooth?

- Wireless standard for exchanging data over short distances.
- Short wavelength UHF: 2.4 – 2.485 GHz
- 79 channels (usually) + Adaptive Frequency Hopping
- Name coming from Harald Bluetooth



- Scandinavian humor... ☺

What the hell is Bluetooth?

Layer protocol architecture

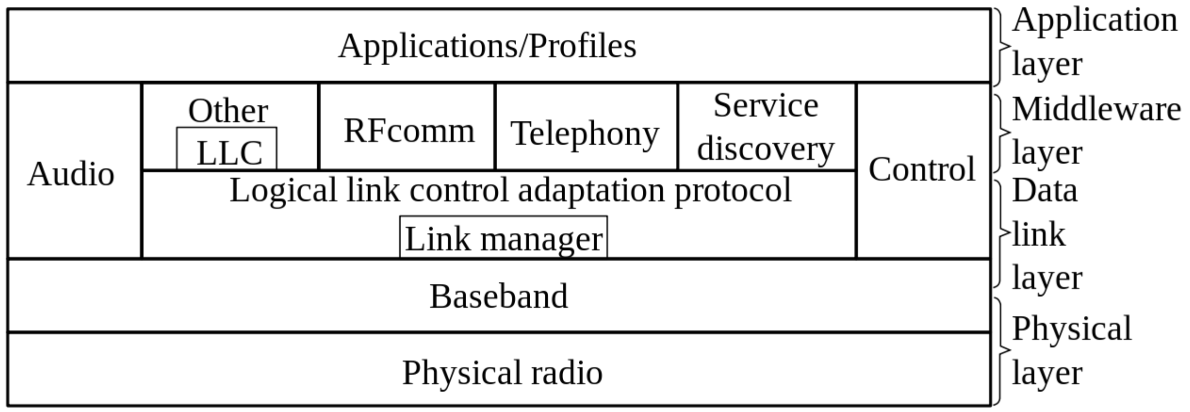
Core protocols

Cable replacement protocols

Telephony control protocols

Adopted protocols

So many different stacks!



LMP, L2CAP, SDP are mandatory!

What the hell is Bluetooth?

- So many updates!

Version 1:

- 1.0: Mandatory BD_ADDR
- 1.1: IEEE Standard (2002)
- 1.2: Adaptive frequency-hopping spread spectrum → resistance to interferences and eavesdropping (theorically ☺)

Version 2:

- 2.0: EDR (optional) for faster data transfer, GFSK+PSK modulation
- 2.1: Secure Simple Pairing, Extended Inquiry Response

What the hell is Bluetooth?

- So many updates!

Version 3: $\longrightarrow$

- 3.0: Alternative MAC/PHYs for high data transfer, Unicast Connectionless Data

Version 4: $\longrightarrow$

- 4.0: Includes now Bluetooth Low Energy protocol (or Smart)
- 4.1: Limited discovery time, lower consumptions, LE link layer topology
- 4.2: LE Data packet extension, LE «secure» connections, Link Layer privacy (really?)

Index

securenetwork
Your Protection, Our Mission

OPPOSING FORCE
challenging your security

Known and unknown risks..

- BlueSnarf, by Holtmann & Laurie

When? → Late 2003

What? → Bluetooth implementation on mobile phones and pocket palms

Why? → «(in)security» of OBEX protocol

Easy GET requests to common files (calendar, contacts..)

No authentication needed

No prompts on the user’s side

securenetwork Your Protection, Our Mission

BlueSnarf™

OPPOSING FORCE challenging your security

Known and unknown risks..

- BlueBug, by Adam Laurie & Martin Herfurt

When? $\longrightarrow$ 2004 @DEFCON12

What? $\longrightarrow$ Bluetooth implementation on mobile phones, especially Symbian OS

Why? $\longrightarrow$ Security loophole

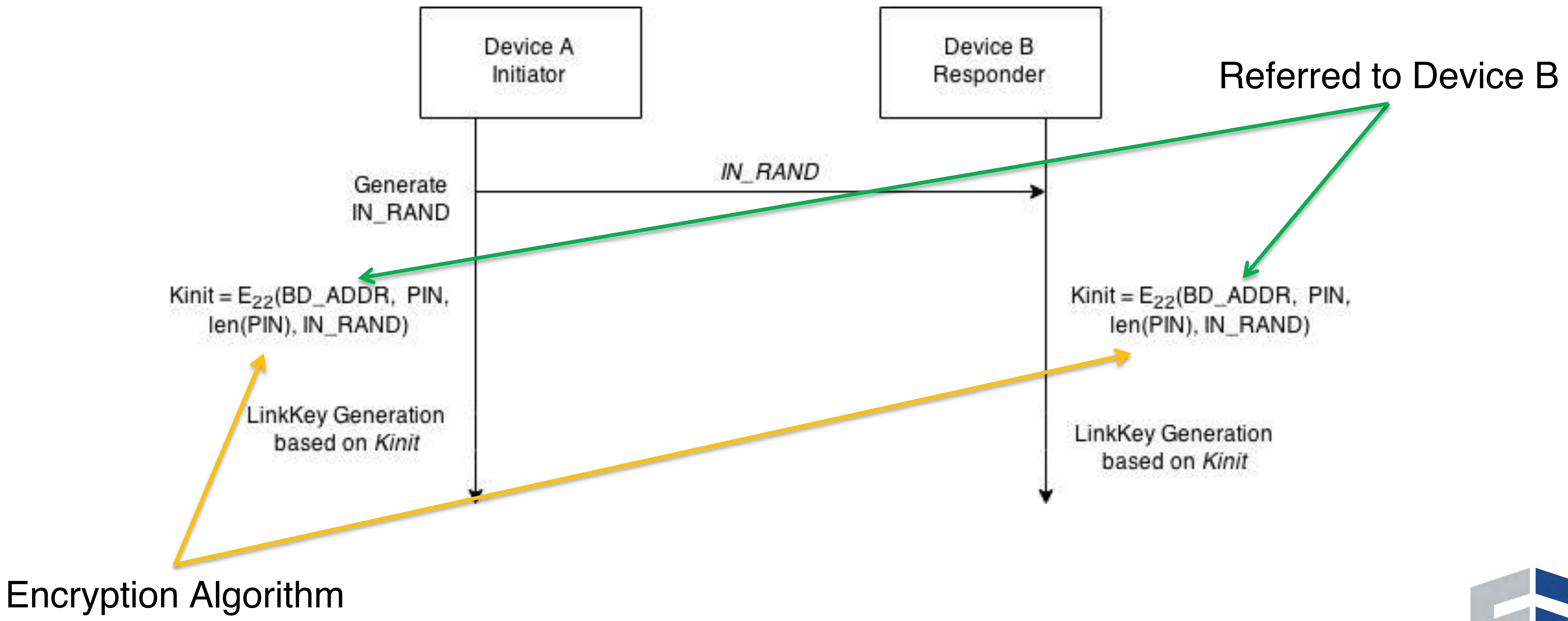
- Control device thorugh plain serial connection
- Download items via OBEX protocol w/out prompts
- No secure auth prior to v2.0

securenetwork Your Protection, Our Mission

OPPOSING FORCE challenging your security

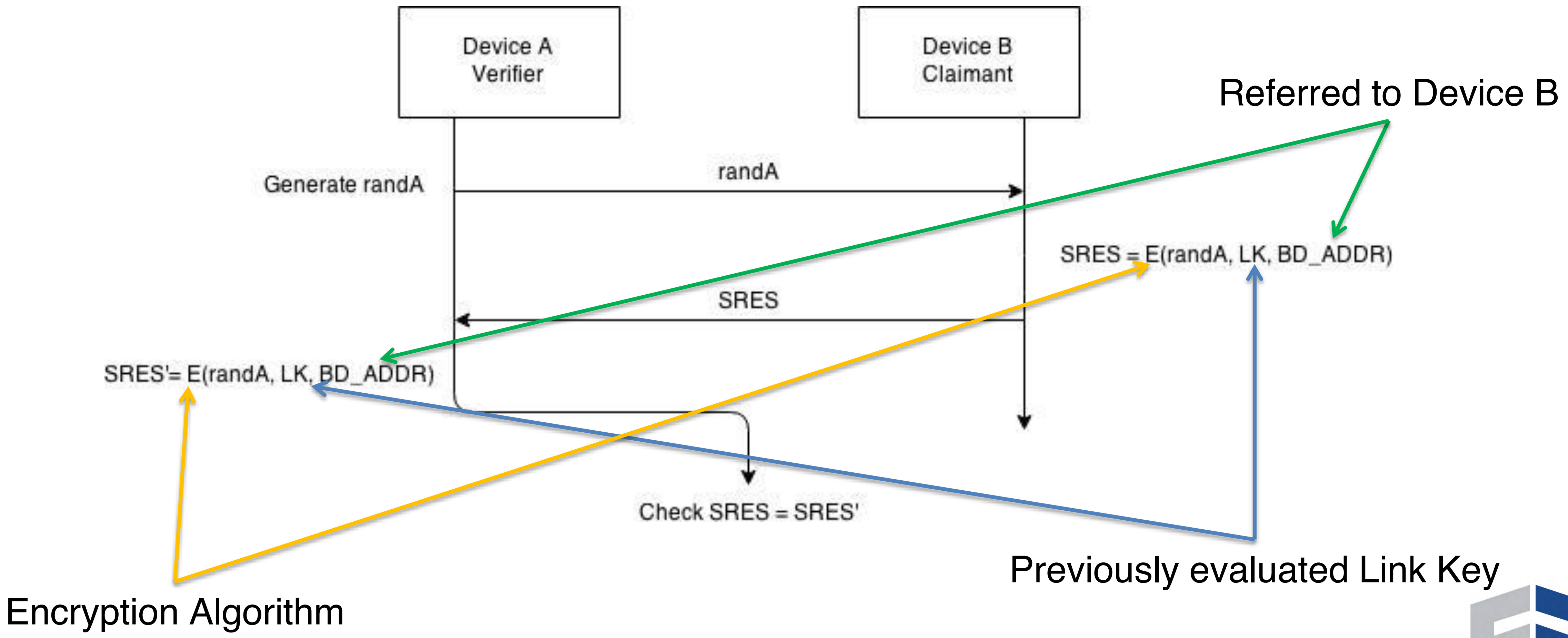
Known and unknown risks..

- Legacy (prior to v2.0) pairing procedure


Device A
Initiator
Device B
Responder
Referred to Device B
IN_RAND
Generate IN_RAND
Kinit = E22(BD_ADDR, PIN, len(PIN), IN_RAND)
Kinit = E22(BD_ADDR, PIN, len(PIN), IN_RAND)
LinkKey Generation based on Kinit
LinkKey Generation based on Kinit
Encryption Algorithm

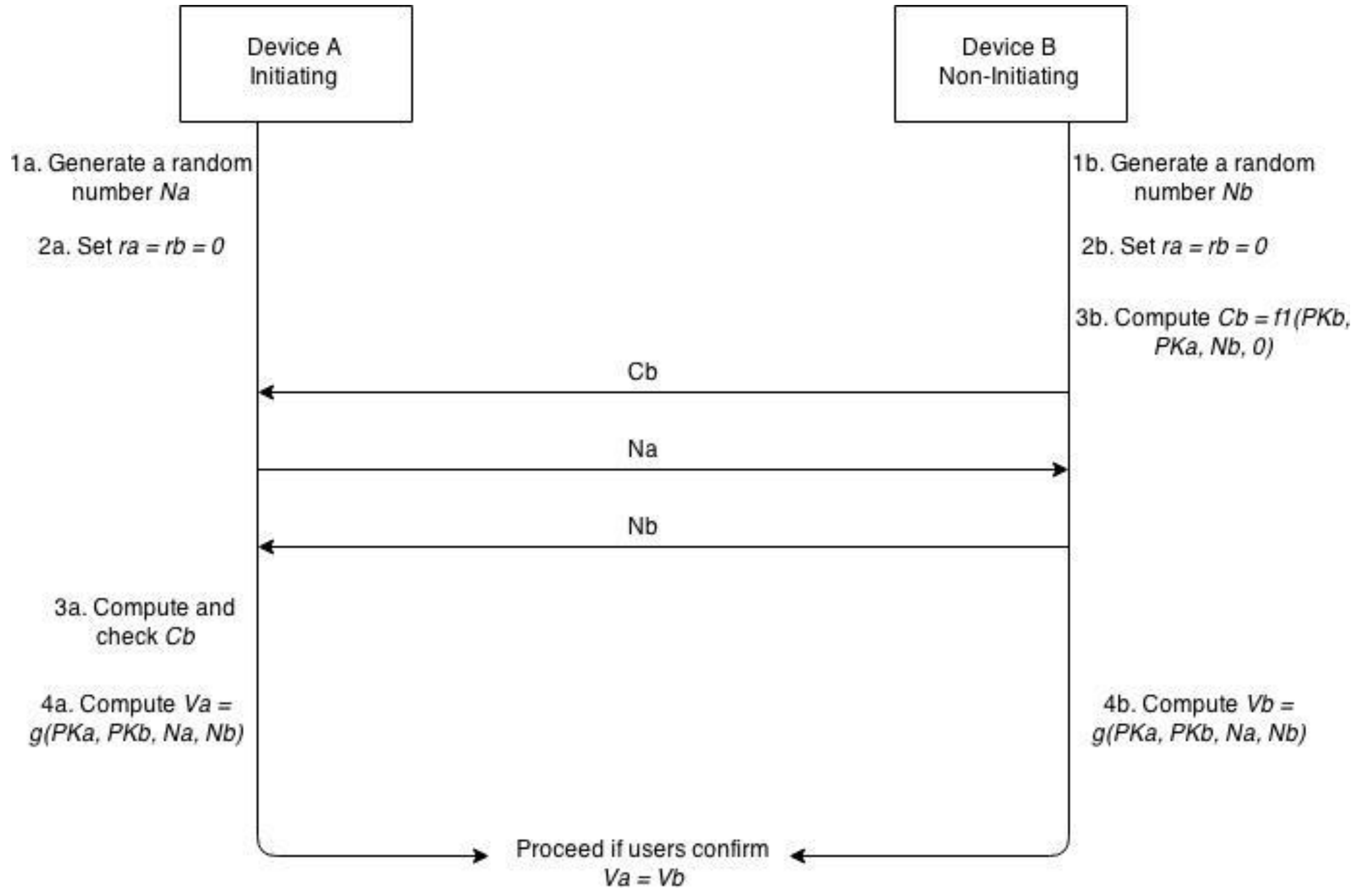
Known and unknown risks..

- Legacy (prior to v2.0) authentication procedure



Known and unknown risks...

- Secure simple pairing



Device A Initiating
Device B Non-Initiating
1a. Generate a random number Na
1b. Generate a random number Nb
2a. Set ra = rb = 0
2b. Set ra = rb = 0
3b. Compute Cb = f1(PKb, PKa, Nb, 0)
Cb
Na
Nb
3a. Compute and check Cb
4a. Compute Va = g(PKa, PKb, Na, Nb)
4b. Compute Vb = g(PKa, PKb, Na, Nb)
Proceed if users confirm Va = Vb

Known and unknown risks..

- BlueChop, following BlueSnarf

What? → It disrupts any bluetooth piconet from the outside

Provided → Master must support multiple connections

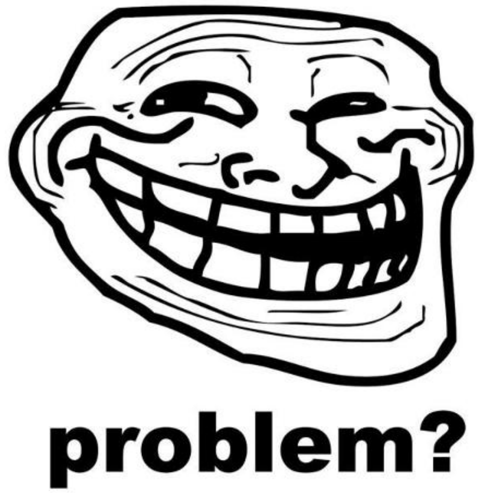
Spoof a random slave out of the piconet → Contact the master → Confusion of the master’s internal state → Piconet disruption ☺

Known and unknown risks..

- Bluetooth LE encryption bypass, by Mark Ryan:
  - Eavesdropping vs Decrypting
  - 3 different keys needed to estabilish a connection, TK, STK, LTK
  - If we are able to save the key exchange procedure, we are done ☺

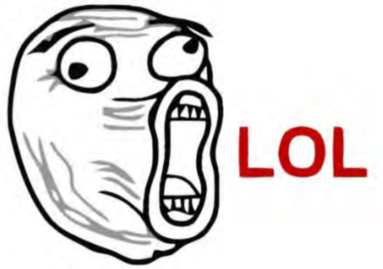
$$\begin{tikzcd} \text{Pairing} \arrow[dr] & & \text{STK} \arrow[dr] & \\ & \text{TK} \arrow[ur] & & \text{LTK} \end{tikzcd}$$

- What if I get TK?



Known and unknown risks..

- TK, 128 bit AES key, depends on the pairing mode:



Just Works

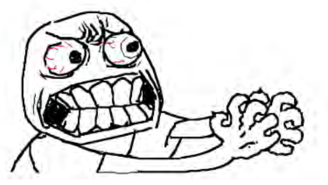
TK = 0

6-digit PIN

TK = 128-bit number

Out Of Band (OOB)

TK = #fuckyourself



- Bruteforce is the way. Intel i7, just one core → less than 1 sec

$$\begin{tikzcd} \text{TK} \arrow[r] & \text{STK} \arrow[r] & \text{LTK} \arrow[r] & 42 \end{tikzcd}$$

- The whole procedure may be computed offline

SmartUnlock…

Officially introduced with Android 5.0 it enables to unlock the smartphone without user interaction if at least one of the following conditions apply:

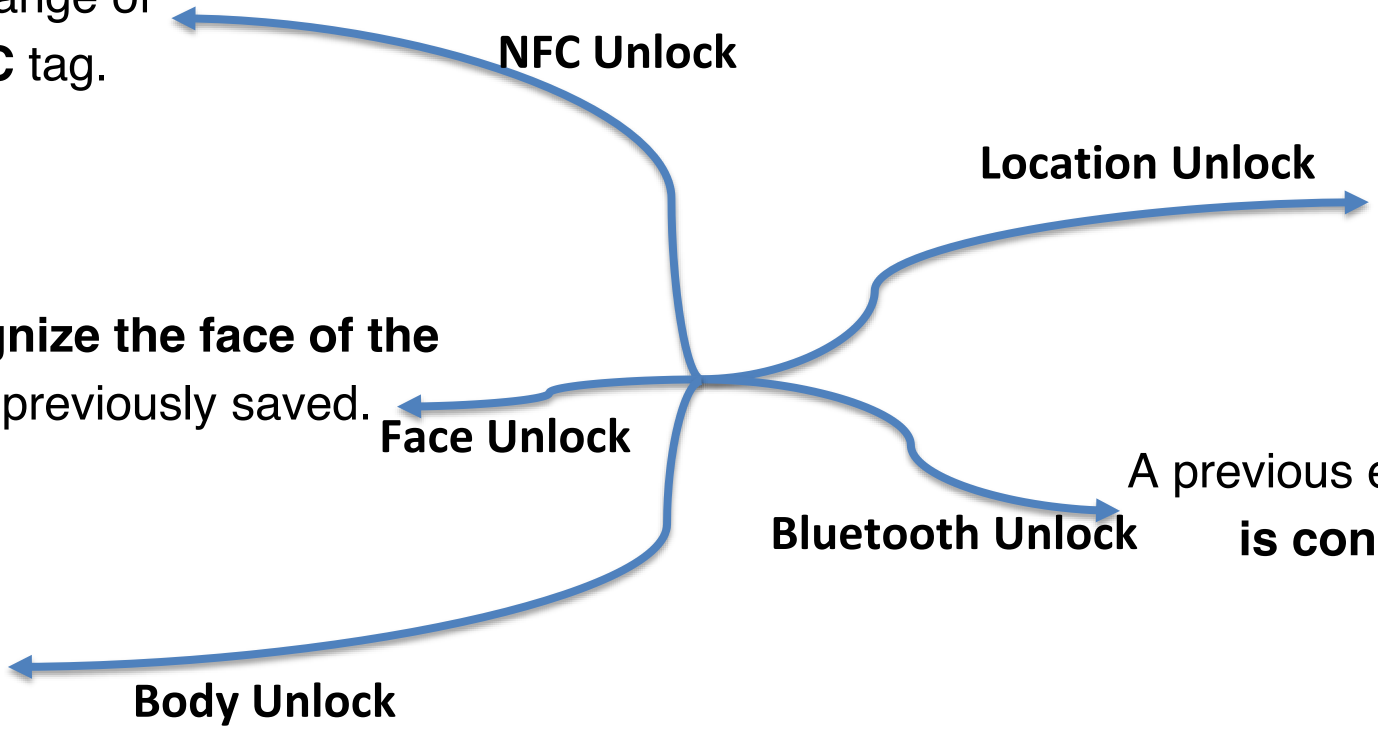
The smartphone is in range of a **previous saved NFC** tag.

The smartphone is **within a certain location**.

The smartphone **recognize the face of the owner**, which must be previously saved.

A previous enabled **bluetooth device is connected** to the smartphone

The smartphone is **in contact with a body**.



Bluetooth Unlock…

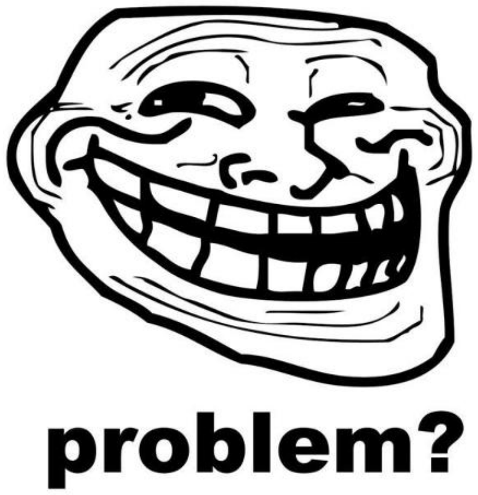
Bluetooth Unlock

This may be the most interesting and most used function of all the above.

The user set a paired bluetooth device as Trusted, and from now on every time that device is linked to the smartphone the lockscreen is bypassed.

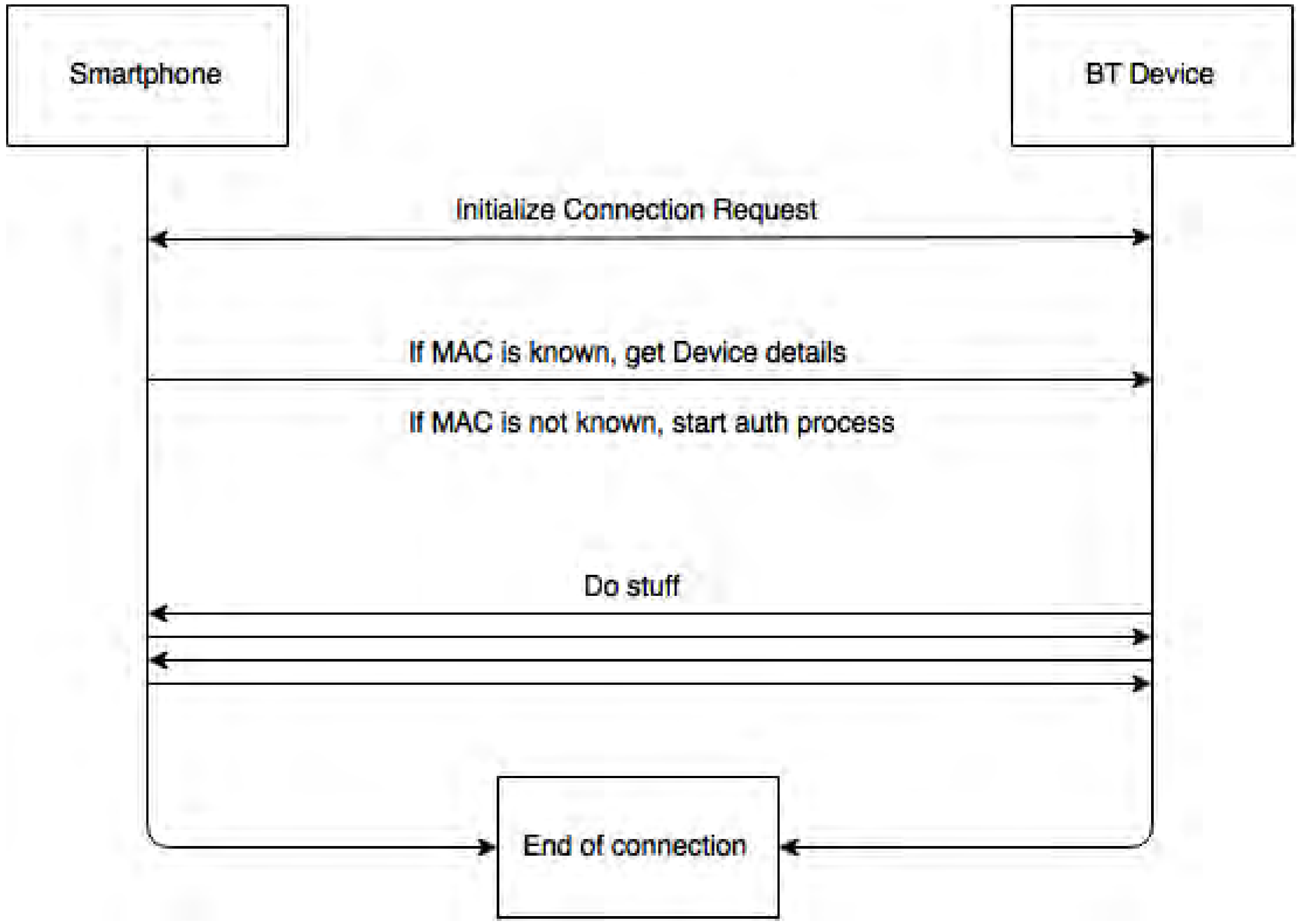
Good, so what is the problem?

Bluetooth Unlock…



In Android < 5.1 the LK (LinkKey) **is not checked** to verify the Bluetooth device.

Bluetooth Unlock…


Smartphone
BT Device
Initialize Connection Request
If MAC is known, get Device details
If MAC is not known, start auth process
Do stuff
End of connection

Bluetooth Unlock…

Now the question is:

How to get the 4 bytes of the MAC address required?

Two possible solutions:

Bruteforce

- Slow
- Expensive
- Not such a good idea

Sniffing

- Requires vicinity
- Target can become aware
- Authentication process is required

Bruteforce…

- **Slow** → { We cannot bruteforce the MAC address offline, we need to try a new connection everytime
- **Expensive** → { We can speed it up parallelizing it but costs increase.
- **Not such a good idea** → { 42 bits will defentely requires too much time.

Sniffing…

- **Requires vicinity** $\rightarrow$ { Target must be near enough for our ubertooth to intercept packets
- **Target can become aware** $\rightarrow$ { Target can be suspicious of strange guy with big antenna(s)
- **Auth process is required** $\rightarrow$ { Usually only 3 bytes of MAC address are transmitted

Our approach…

Hybrid is always the **solution**

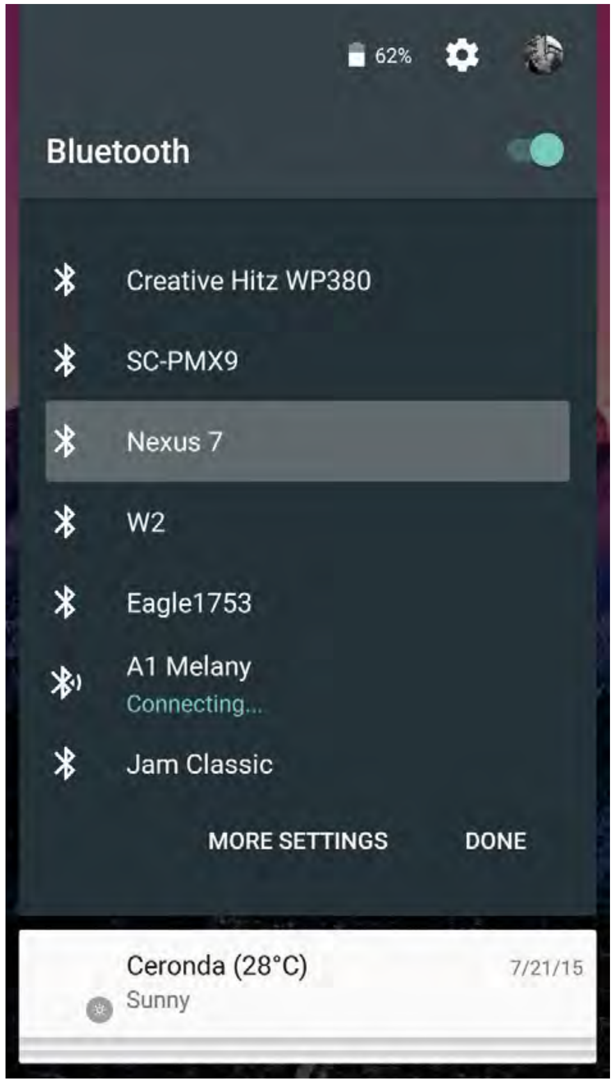
- Android automatically sends out ‘beacons’ of paired BT devices.
- The trusted device **must** be a paired device
- We can intercept beacons to retrive 3 bytes of the MAC address
- Bruteforce the remaining… 1 bytes = 256 possible MAC addresses

Demo Time!

<video demo>

New findings...

Android 5.1 adds a new nice feature...


62%

Bluetooth

Creative Hitz WP380

SC-PMX9

Nexus 7

W2

Eagle1753

A1 Melany
Connecting...

Jam Classic

MORE SETTINGS DONE

Ceronda (28°C) 7/21/15
Sunny

Demo Time!

<video demo>

New findings...

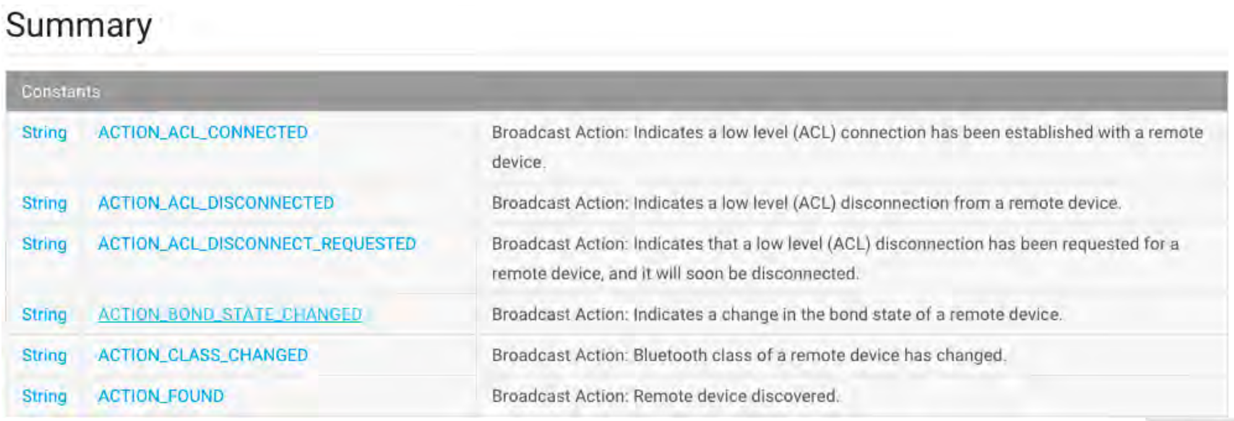
Is it fixed?

It depends…

Android >= 5.1
SmartUnlock is fixed
API are still vulnerable

Android <= 5.0.X
SmartUnlock is not fixed
API are vulnerable

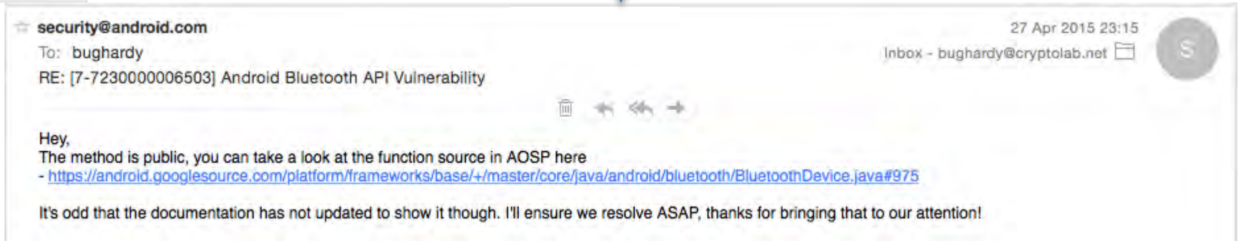
New findings...



Summary

Constants		
String	ACTION_ACL_CONNECTED	Broadcast Action: Indicates a low level (ACL) connection has been established with a remote device.
String	ACTION_ACL_DISCONNECTED	Broadcast Action: Indicates a low level (ACL) disconnection from a remote device.
String	ACTION_ACL_DISCONNECT_REQUESTED	Broadcast Action: Indicates that a low level (ACL) disconnection has been requested for a remote device, and it will soon be disconnected.
String	ACTION_BOND_STATE_CHANGED	Broadcast Action: Indicates a change in the bond state of a remote device.
String	ACTION_CLASS_CHANGED	Broadcast Action: Bluetooth class of a remote device has changed.
String	ACTION_FOUND	Broadcast Action: Remote device discovered.

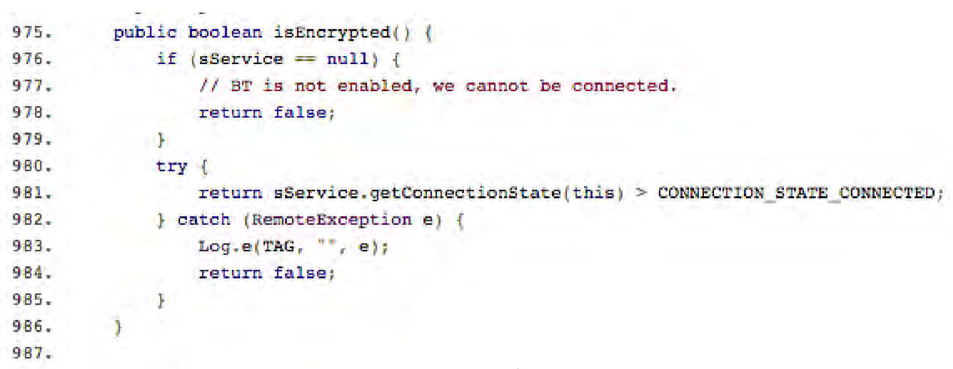
API does not have a safe method to check if a device is connected with a proper LK



security@android.com 27 Apr 2015 23:15
To: bughardy Inbox - bughardy@cryptolab.net
RE: [7-7230000006503] Android Bluetooth API Vulnerability

Hey,
The method is public, you can take a look at the function source in AOSP here
- https://android.googlesource.com/platform/frameworks/base/+/master/core/java/android/bluetooth/BluetoothDevice.java#975

It's odd that the documentation has not updated to show it though. I'll ensure we resolve ASAP, thanks for bringing that to our attention!



```
975.     public boolean isEncrypted() {
976.         if (sService == null) {
977.             // BT is not enabled, we cannot be connected.
978.             return false;
979.         }
980.         try {
981.             return sService.getConnectionState(this) > CONNECTION_STATE_CONNECTED;
982.         } catch (RemoteException e) {
983.             Log.e(TAG, "", e);
984.             return false;
985.         }
986.     }
987.
```

Android Security Team told us that there is a method for this, but it was not yet in SDK, as 27th April, 2015. And it still not present

Demo Time!

Why fixing the API is important if SmartUnlock function is fixed?

↓

3rd party applications!

↓

Demo time!

Index

{

secure network
Your Protection, Our Mission

OPPOSING FORCE
challenging your security

Future Works...

Bluetooth is everywhere, we are focusing on:

- IoT Devices
- Smart Locks
- Fit Band
- etc

Thank you

Q&A Time…