
# Bluetooth Hacking revisited


Kevin Finistere & Thierry Zoller

23C3 - 2006

# Bluetooth – Please just turn it off

Turn off your BT please,


,no really.

# The Goal of this Talk ?

- The Goal of this talk is not to:
  - Build myths
  - Show off – and not show how

- The Goal of this talk is to :
  - Raise awareness
  - Make risks (more) transparent
  - Paradigm Shift – Bluetooth is not only for toys
  - Show cool stuff…

# What are we talking about today ?

- **[ 0x00 ] – Introduction : What is Bluetooth ?**
  - Sorry this is required. Crash course..


- **[ 0x01 ] – Get ready to rumble : Extending the Range**
  - Extending the range of Bluetooth devices
  - Building automated reconnaissance and attack devices
  - Bluetooth War driving (GPS, 360° Camera)

- **[ 0x02 ] – Implementation issues : Bypassing Security**
  - Attacking drivers, Attacking applications
  - Owning Bluetooth VNC style
  - Attacking Internal Networks and pivoting
  - Bluetooth Pin to Bluetooth Passkey

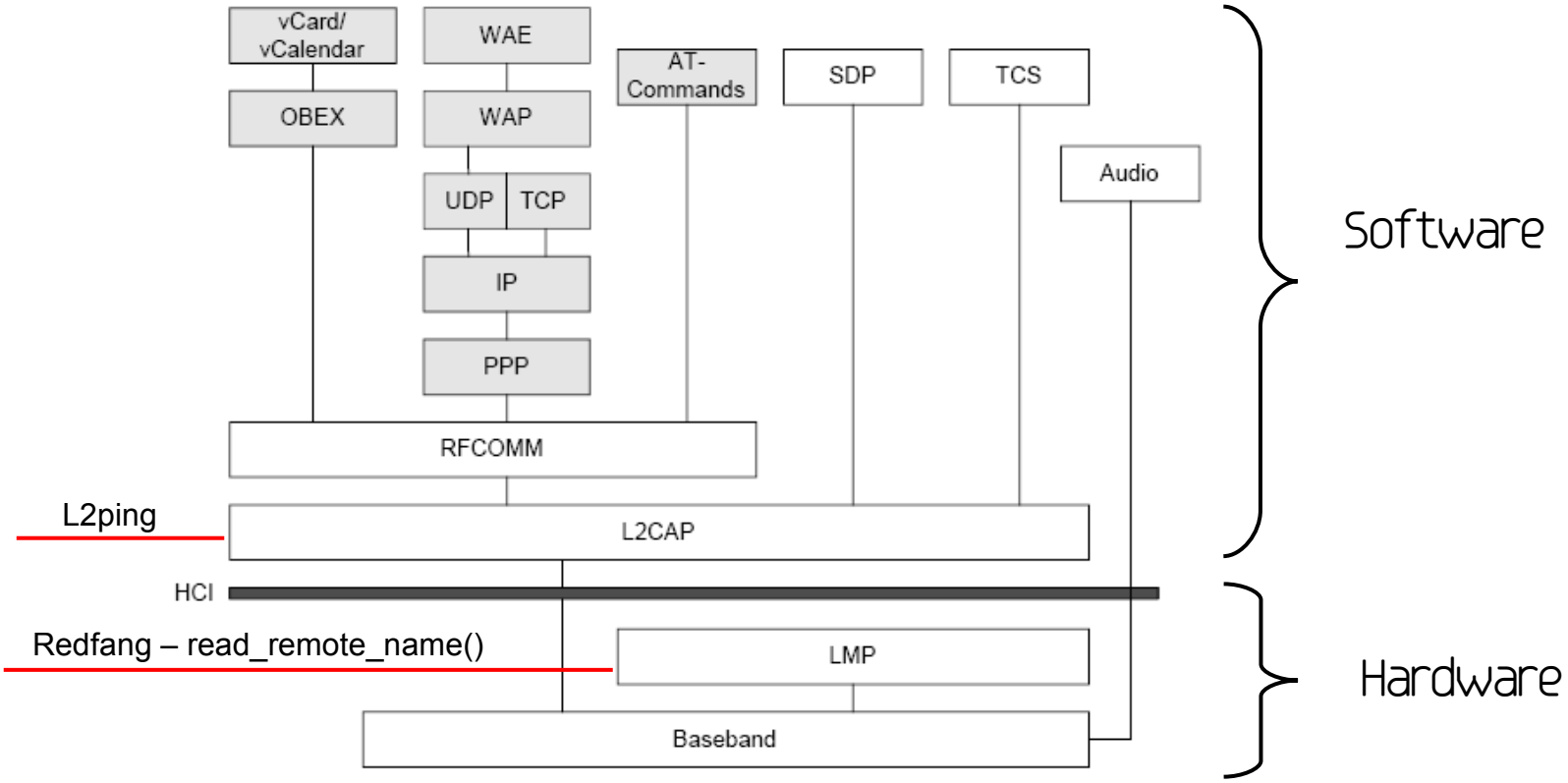
- **[ 0x03 ] – Protocol/Specification issues : Ceci n'est pas une pipe**
  - Cracking the Pin and the Link-key (BTCrack)
    - Key management, 8 bit Encryption, Collisions
  - Tracking the un-trackeable
  - Anti-Brute-forcing
  - Random Number generators from hell

# [ 0x00 ] Introduction

- Bluetooth - a few tidbits:
  - Operates on the non-regulated ISM band : 2,4Ghz
  - In general 79 Channels (Except France, Spain)
  - Frequency Hopping (3200/sec, 1600/sec)
  - Complete Framework with profiles and layers of protocols
  - 1 Billionth BT device sold in November 2006 (source SIG)
  - Goals : Least cost cable replacement, low power usage

# [ 0x00 ] Introduction

- The foundation – Protocol Stack



# [ 0x00 ] Introduction

- "Typical" Bluetooth Scenario

Bluetooth
Access Point

# [ 0x00 ] Introduction

- Inquiry - First Contact
  - Predefined Hopping sequence
  - FHS same for all devices
  - Pass Paging parameters during Inquiry stage

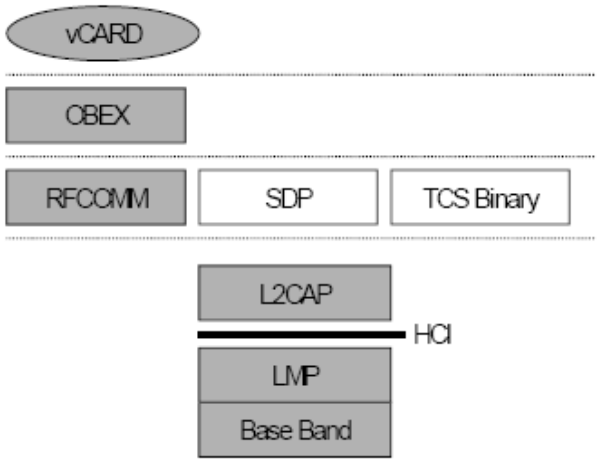
# [ 0x00 ] Introduction

- Paging - Frequency Hopping Synchronization
  - Slaves always sync to the Master
  - Paging initialisation :
    - Slaves hop 1 Channel/sec
    - Master hops 3200 times/sec
  - Paging
    - Both hop 1600 times/sec
    - Piconet agrees to a Sequence based on parts of the BD_ADDR and Clock-offset of the master. (Nice fingerprint by the way)

  - FH is the reason you can not easily sniff BT traffic. You have to sync to the Master (or use a Spectral Analyzer and reconstruct afterwards – Good luck)

# [ 0x00 ] Introduction

- The Bluetooth Profiles
  - Represent a group and defines mandatory options
  - Prevent compatibility issues, modular approach to BT extensions
  - Vertical representation of BT layer usage, handled through SDP




Object Push Profile

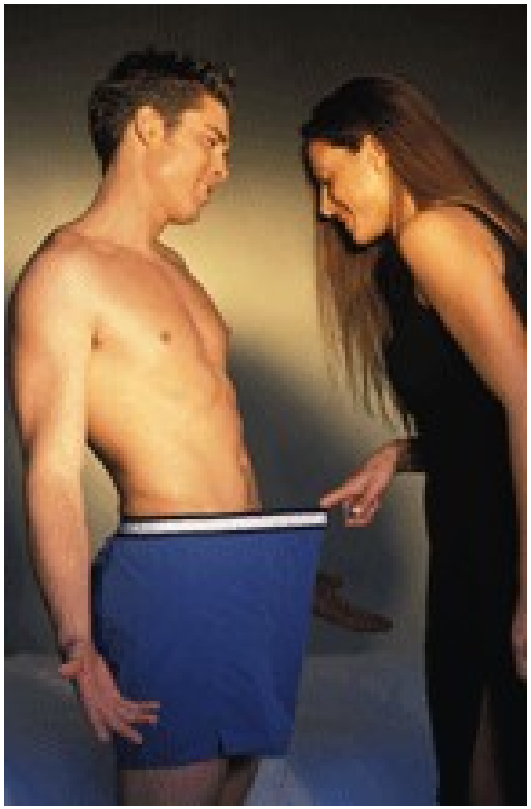
```
Service Name: OBEX Object Push
Service RecHandle: 0x10001
Service Class ID List:
  "OBEX Object Push" (0x1105)
Protocol Descriptor List:
  "L2CAP" (0x0100)
  "RFCOMM" (0x0003)
    Channel: 9
  "OBEX" (0x0008)
Language Base Attr List:
  code_ISO639: 0x656e
  encoding:    0x6a
  base_offset: 0x100
Profile Descriptor List:
  "OBEX Object Push" (0x1105)
    Version: 0x0100
```

# [ 0x00 ] Introduction

- Different Bluetooth modes
  - Discoverable modes
    - **Discoverable** :
      Sends inquiry responses to all inquiries.
    - **Limited discoverable**:
      Visible for a certain period of time (Implementation bug: Sony Ericsson T60..)
    - **Non-Discoverable**:
      Never answers an inquiry scan (in theory)
  - Pairing modes :
    - **Non-pairable mode :**
      Rejects every pairing request (LMP_not_accepted) (Implementation bug: Plantronic Headset..)
    - **Pairable mode :**
      Will pair up-on request

# [ 0x01 ] Get ready to rumble

- Extending the Range



# [ 0x01 ] Get ready to rumble

- Long Distance - Datasets
  - Antrum Lake, water reflection guarantees longer ranges.
  - 788 Meters
  - An old Man stole my phone during this test! I tracked him with the yagi.

# [ 0x01 ] Get ready to rumble

- Optimizing for Penetration (1)
  - Integrated Linksys Dongle
  - Integrated USB Cable
  - Metal Parabola
  - 10 * Zoom
  - Laser (to be done)


- Experiment : Went through a building found the device on the other side IN another building.

# [ 0x01 ] Get ready to rumble

- Optimizing for Penetration (2)
  - Bundling (Parabola)
  - Higher penetration through walls
  - Glass is your friend
  - On board embedded device. (NSLU2)
  - Autonomous scan and attack toolkit
    - automatically scans
    - may attack devices
    - saves all the results

# [ 0x01 ] Get ready to rumble

- PerimeterWatch – Bluetooth Wardriving
  - Perl Script by KF
  - Searches Bluetooth Devices
  - Takes 360° pictures
  - GPS coordinates

# [ 0x02 ] Implementation bugs

- Implementation Bugs – Bypassing security

# [ 0x02 ] Implementation bugs

- Menu du Jour :
  - Eavesdropping on Laptops/Desktops
  - Remotely controlling workstations
  - Car Whisperer NG
  - Owning internal Networks over Bluetooth
  - Linkkey theft and abuse
  - Widcomm Overflows
    (Broadcom merger leaves lots of vuln users that can not patch) BTW 3.0.1.905 (../ attacks) and up to BTW 1.4.2.10 has overflows

# [ 0x02 ] Implementation bugs

- Bluetooth PIN is really a Bluetooth Passkey
  - Did you know ? A Bluetooth “Pin” can be more than digits…
  - Not aware of any implementation, all use just digits
  - Uses UTF8
  - Max 16, UTF8 char may take some off
  - Example :

| User enters | BT handles |
|---|---|
| 0123 | 0x30 0x31 0x032 0x33 |
| Ärlich | 0xC3 0x84 0x72 0x6c 0x69 0x63 0x68 |

  - It’s like implementing NTLM with digits only….
  - BTCrack would a lot more time if this would be “correctly” implemented

# [ 0x02 ] Implementation bugs

- CarWhisperer – Martin Herfurt
  - Listen and Record Conversations
  - Not that new, but what's new :
    - **Works against Workstations**
      Example : Widcomm < BTW 4.0.1.1500 (No Pincode)
    - Kevin did a real-time patch for it
    - Remove the Class ID check
  - Root Cause :
    Paring mode, discoverable, hard coded Pin.

```
SWITCH: for ($bdaddr) {
        /00:02:EE/          && do { $pin="5475"; last;}; # Nokia
        /00:0E:9F/          && do { $pin="1234"; last;}; # Audi UHV
        /00:80:37/          && do { $pin="8761"; last;}; # O'Neill
        /00:0A:94/          && do { $pin="1234"; last;}; # Cellink
        /00:0C:84/          && do { $pin="1234"; last;}; # Eazix
        $pin="0000"; # 0000 is the default passkey in many cases
        }
```

# [ 0x02 ] Implementation bugs

- HidAttack - Owning Bluetooth VNC Style
  - HID = Human Interface Device
  - Requires 2 HID (PSM) endpoints to act as server
  - 2 implementations :
    - Keyboard connects to the HID server
    - HID server connects to the Keyboard
  - You can control the Mouse and Keyboard HID just as you were in front of the PC.

  - Discovered by **Collin Mulliner** , fixed in hidd Bluez <2.25, Widcomm, Toshiba not really tested. Yours?
  - Code release today : www.mulliner.org/bluetooth/hidattack01.tar.gz
  - Thanks **Collin** !

# [ 0x02 ] Implementation bugs

- Demo - Owning internal networks
  - Apple
    - OSX 10.3 Tiger
    - OSX 10.4 Jaguar
      Vanilla, delayed release
  - Windows
    - Widcomm, Toshiba,
      Bluesoil, others ?
  - Pocket PC

- Kevin: Apple asked me to not tell 10.4 was shipping vulnerable
- OSX 10.3.9 patched, OSX 10.4 shipped vulnerable **patched a month after** OSX 10.3.9

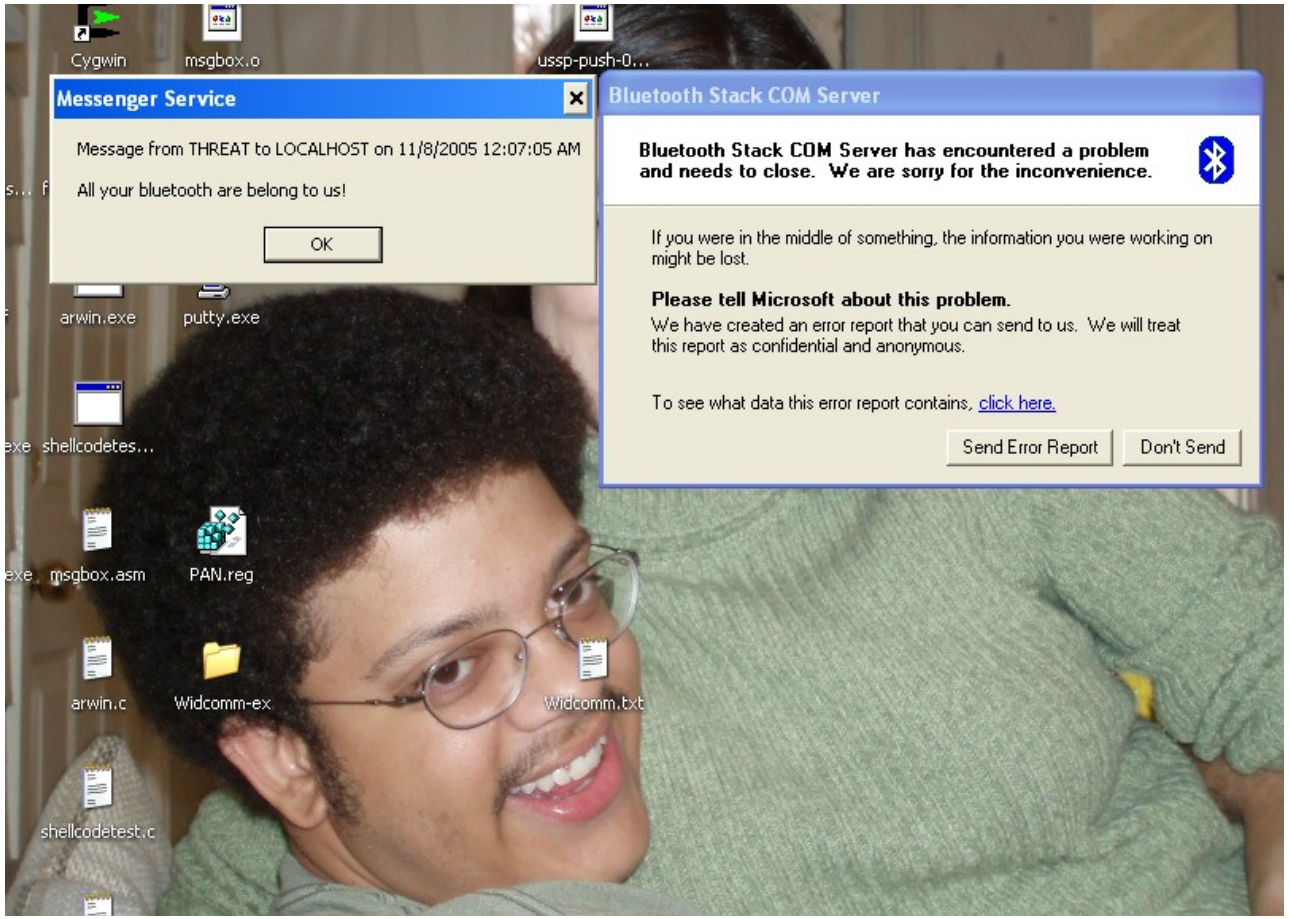
# [ 0x02 ] Implementation bugs

- Demo – Remote Root over BT
  - Vulnerability shown :
    _Directory Traversal_ in un-authenticated Obexserver (Patched)
  - Cause :
    User input validated client-side (except btftp)

  - ObexFTP server directory traversal exploit & malicious InputManager & local root exploit = remote login tty over rfcomm = 0WNAGE
  - Was possible on Windows and Pocket PC and everything that has Toshiba or Broadcom & Widcomm (estimate 90%), and most probably others too. But we choose a MAC, because…we can.
  - Points are :
    - Macs are NOT invulnerable (far from that) - You can own internal networks over Bluetooth

# [ 0x02 ] Implementation bugs

- Windows Widcomm - Buffer overflows



# [ 0x02 ] Implementation bugs

- Windows Widcomm - Buffer overflows
  - Vulnerable versions known to us :
    - Widcomm Stack up to 3.x is vuln
    - Widcomm BTStackServer 1.4.2 .10
    - Widcomm BTStackServer 1.3.2 .7
    - Widcomm Bluetooth Communication Software 1.4.1 .03
    - HP IPAQ 2215
    - HP IPAQ 5450

# [ 0x03 ] Protocol issues

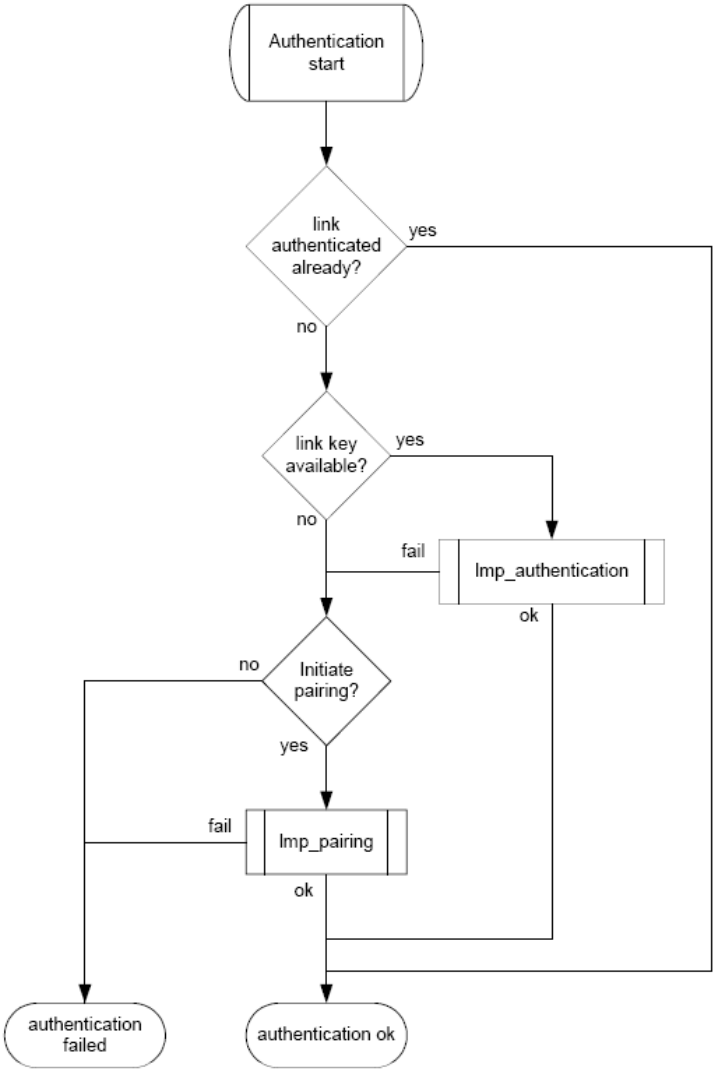

*This is supposed to be a joke

# [ 0x03 ] Protocol issues

- Menu du Jour :
  - Why the Pin is not that important
  - Unit Keys
  - How to find non discoverable devices
  - Random Number generators that may be from Hell
  - Link Keys
    - Reconstructing them
    - Abusing them
    - Re-force Pairing, Corruption
  - Denial of Service

# [ 0x03 ] Protocol issues

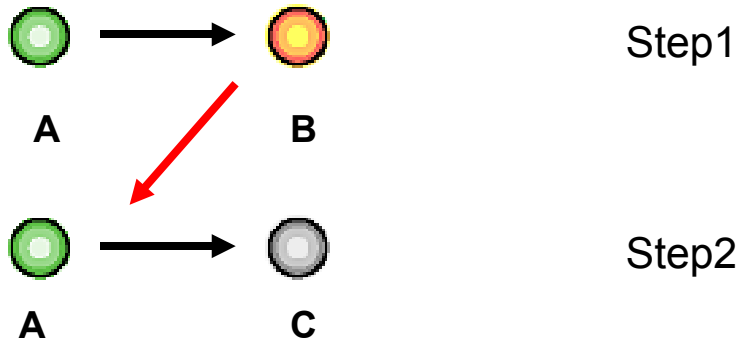
- The PIN is not really that useful
  - The link key is !
  - Here's why :
    - Pairing mode required for PIN
    - The LK is enough to authenticate
    - Encryption (E0) calculated from the LK
    - We can authenticate against both sides with the same key




- Protocol 1.2 Authentication :

# [ 0x03 ] Protocol issues

- Unit keys
  - Generated by the device when starting up
  - Based on a PRNG that may come from hell
  - Permanently saved and cannot be changed
  - Only has one key
    - Problem :




  - The SIG clearly does not recommend it’s use.

# [ 0x03 ] Protocol issues

- How to find nondiscoverable devices **passively**
  - From the man himself: Joshua Wright
  - We knew read_remote_name(), now l2ping.
  - Target : BD_Addr : 48-bit
    00:11:9F:C5:F1:AE

4. Sniff on a preset channel and wait for devices to hop by , capture the Bluetooth Preamble, extract the cannel access code (which is based on 24 bits of the BD_addr)
5. Extract Error Correction field (baseband header – CRC 10bit field)
6. Assume the first 8 bits 00
7. Brute force the remaining: 8bits

# [ 0x03 ] Specification issues

- Random Number Generators from Hell
  - Specification is not very clear about what to achieve or how to achieve it
  - The specification reads :

Each device has a pseudo-random number generator. Pseudo-random numbers are used for many purposes within the security functions − for instance, for the challenge-response scheme, for generating authentication and encryption keys, etc.

Within this specification, the requirements placed on the random numbers used are non-repeating and randomly generated

For example, a non-repeating value could be the output of a counter that is unlikely to repeat during the lifetime of the authentication key, or a date/time stamp.

# [ 0x03 ] Specification issues

- Random Number Generators from Hell
  - Remember the Clock inside each Device ?
  - Remember that we can get the clock-offset with an simple non-authenticated inquiry ?
  - RND do not look very random, had no time left to investigate fully, looks horrible.

- They don't trust it themselves :

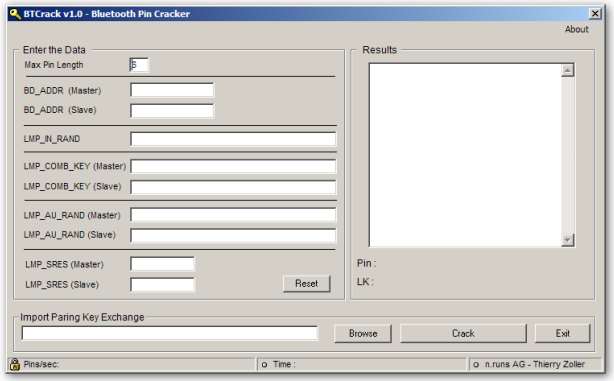
The reason for using the output of and not directly choosing a random number as the key*, is to avoid possible problems with degraded randomness due to a poor implementation of the random number generator within the device.

$$K_{master} = E_{22}(\text{RAND1}, \text{RAND2}, 16).$$

*What a great idea that would have been…

# [ 0x03 ] Protocol issues

- Introducing BTCrack
  - First presented at Hack.lu 2006
  - Released for 23C3
  - Cracks PIN and Link key
  - Requires values from a Pairing sniff
  - Imports CVS Data




Available for download here now:
http://www.nruns.com/security_tools.php

# [ 0x03 ] Protocol issues

- History
  - Ollie Whitehouse - 2003
    - Presents weaknesses of the pairing process and how it may be used crack the PIN
  - Shaked and Wool - 2005
    - Implemented and optimised the attack
    - Found ways to re-initiate pairing
  - Thierry Zoller – 2006
    - Win32 implementation, first public release
    - Tremendous help from somebody that will recognize himself

# [ 0x03 ] Protocol issues

- Speed - Dual-Core P4-2GHZ
  - BTcrack v0.3 (Hack.lu)
    - 22.000 keys per second
  - BTcrack v0.5
    - 47.000 keys per second
  - BTcrack v1.0
    - Thanks to Eric Sesterhenn
      - Optimised for caching, cleaning code, static funcs, removing Junk
      - ICC
    - 185.000 keys per second

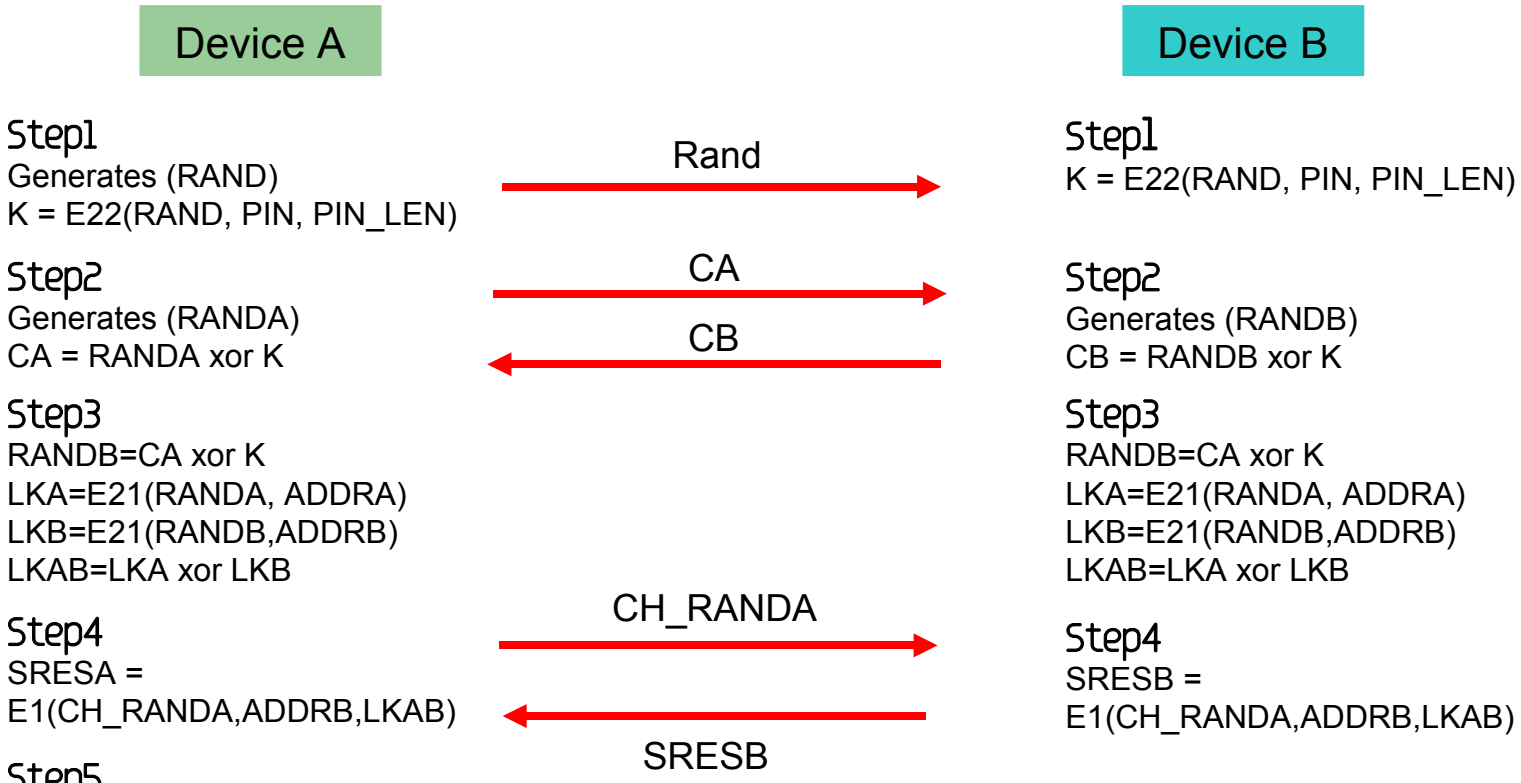
Results :

- 4 digit pin : 0.035 seconds
- 5 digit pin : 0.108 seconds
- 6 digit pin : 4.312 seconds
- 9 digit pin : 1318 seconds

# [ 0x03 ] Protocol issues

- BT Crack – Behind the scenes (1)

E22 = Connection key
E21 = Device key

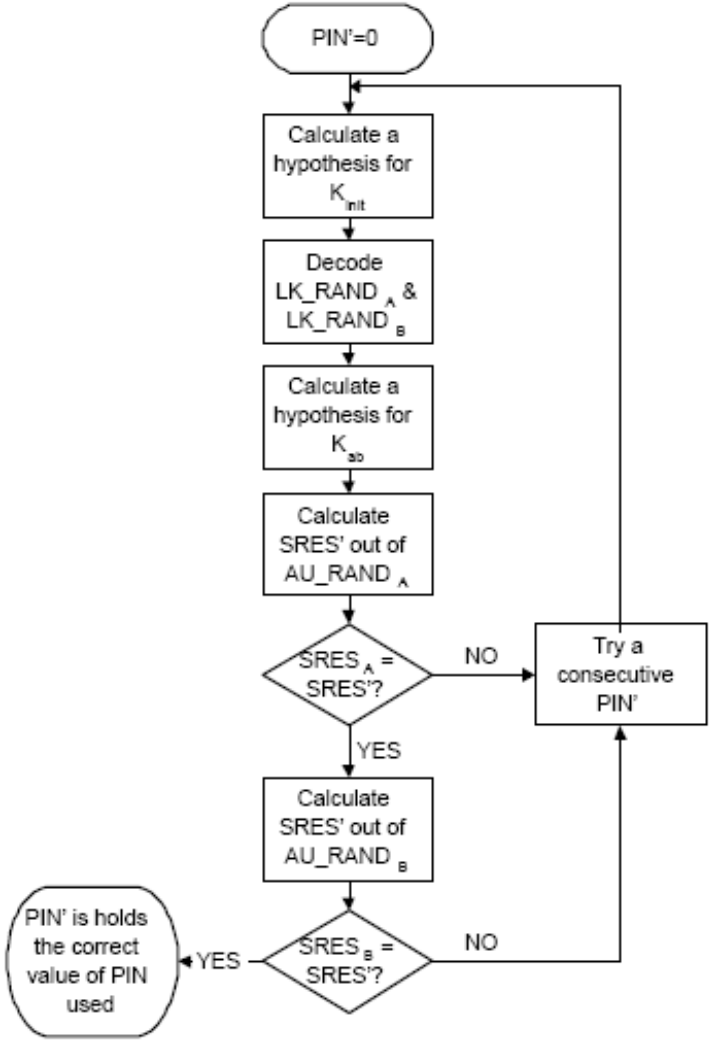


# [ 0x03 ] Protocol issues

- BT Crack – Behind the scenes

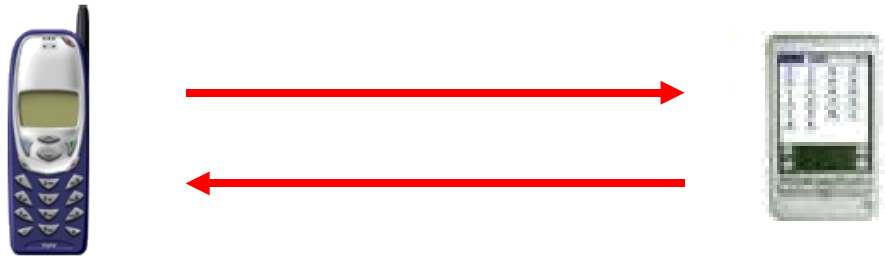
```
Pin =-1;
Do
{
    PIN++;
    CR_K=E22(RAND, PIN, length(PIN));

    CR_RANDA = CA xor CR_K;
    CR_RANDB = CB xor CR_K;

    CR_LKA = E21 (CR_RANDA, ADDRA);
    CR_LKB = E21 (CR_RANDB, ADDRB);

    CR_LKAB = CR_LKA xor CR_LKB;

    CR_SRES = (CH_RAND, ADDRB, CR_LKAB);
}
 while (CR_SRES == SRES)
```

- Right : Shaked and Wool logic
- Top : Pseudo code by Tomasz Rybicki
  Hackin9 04/2005



# [ 0x03 ] Protocol issues

- BT Crack – Demo



# [ 0x03 ] Protocol issues

- Link keys – What can I do with them ?
  - Authenticated to both devices Master & Slave with the same link key
  - Dump them from any Linux, Mac, Windows machine
  - Create a encrypted hidden stealth channel, plant the linkkey
  - You can decrypt encrypted traffic with the linkkey

- How to force repairing ?
  - Shaked and Wool proposed:
    - Injection of LMP_Not_Accepted spoofing the Master
    - Before the master sends Au_rand, inject In_rand to the slave
    - Before the master sends Au_rand, inject random SRES messages
  - We propose :
    - Use bdaddr to change the Bd_Addr to a member, connect to the master with a unknown linkkey.

# [ 0x04 ] Kick-Out

- Sooooo now we have :
  - A quick and reliable way to get the BD_ADDR
  - A way to crack the Pin and the keys

- What's left ?
  - The sniffer. It costs around 13.000$, you can get it on eBay sometimes for the 1/10 of the amount.
  - Assignment : Go and make one for everybody.

# [ 0x04 ] Kick-Out

- Things to Remember :
  - Bluetooth **might** be a risk for your Company
    - Risk assessment is rather complex
  - Don’t accept every file you are being send, just click NO.
  - Disable Bluetooth if not required
  - Pair in “secure” places (SIG Recommendations)
  - Don’t use Unit Keys
  - Hold your Bluetooth vendor accountable for vulnerabilities
  - Delete your pairings
  - Use BT 2.0 and “Simple Paring”