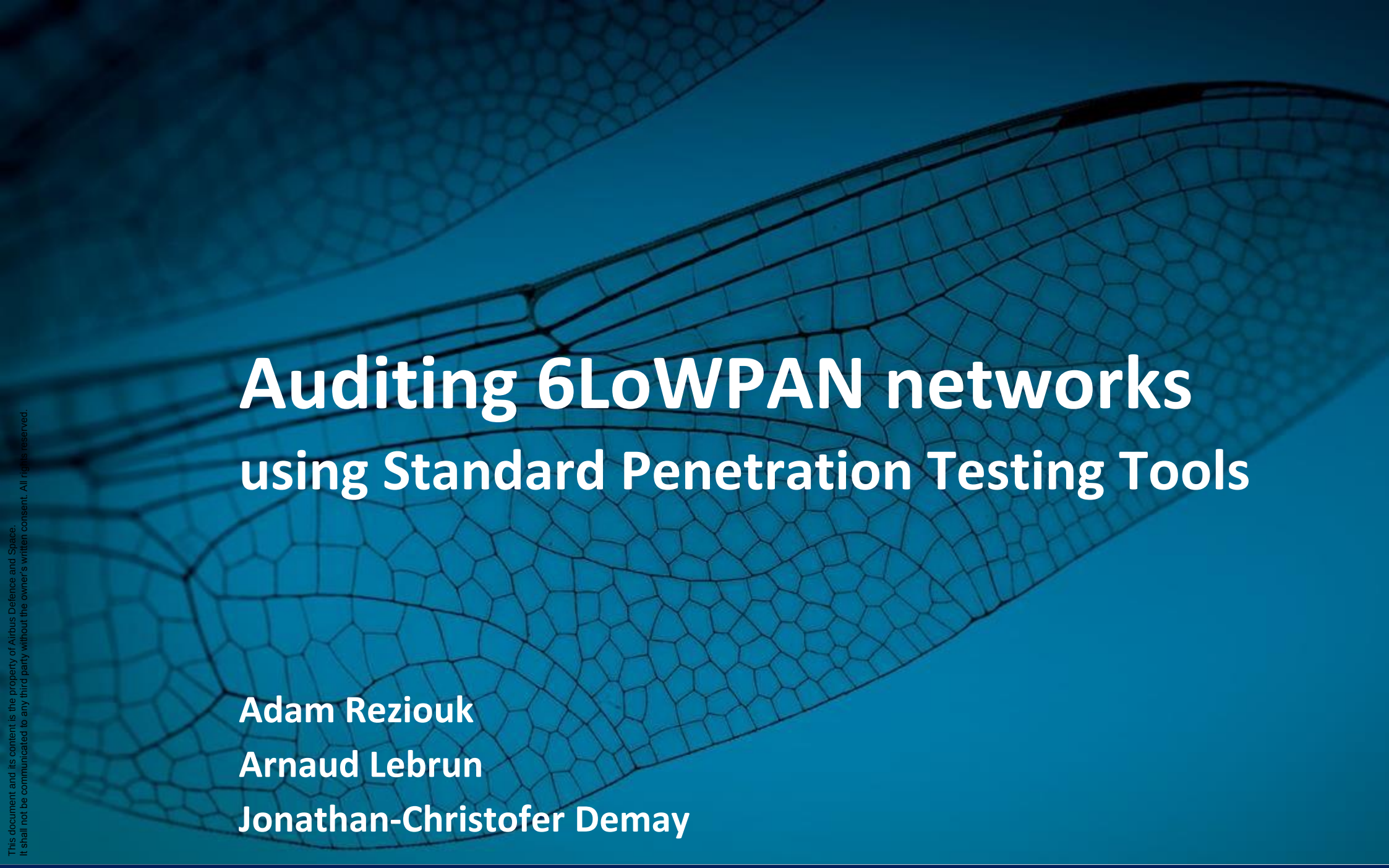


Auditing 6LoWPAN networks
using Standard Penetration Testing Tools

Adam Reziouk
Arnaud Lebrun
Jonathan-Christofer Demay

This document and its content is the property of Airbus Defence and Space.
It shall not be communicated to any third party without the owner's written consent. All rights reserved.

AIRBUS
DEFENCE & SPACE

Presentation overview

- **Why this talk ?**
- **What we will not talk about ?**
- **What we will talk about ?**



The 6LoWPAN protocol

- **IPv6 over Low power Wireless Personal Area Networks**
- **Header compression flags**
  - Addresses factoring (IID or predefined)
  - Predefined values (e.g., TTL)
  - Fields omission (when unused)
  - Use of contexts (index-based)
  - UDP header compression (ports and checksum)
- **Packet fragmentation**
  - MTU 127 bytes Vs 1500 bytes
  - 80 bytes of effective payload

What's the big deal ?

- **Already a lot of tools to work with IPv6**
  - nmap -6, nc6, ping6, etc.
  - Nothing new here !
- **Higher-layer protocols are the same**
  - TCP, UDP, HTTP, etc.
  - Again, nothing new here !
- **Why not use a USB adapter ?**
  - That works for Wi-Fi
  - They are available



The IEEE 802.15.4 standard

- **PHY layer and MAC sublayer**
- **Multiple possible configurations**
  - Network topology: Star Vs Mesh
  - Data transfer model: Direct or Indirect, w/or w/o GTS, w/ or w/o Beacons
- **Multiple security suites**
  - Integrity, confidentiality or both
  - Integrity/Authentication code size (32, 64 or 128)
- **Multiple standard revision**
  - 2003
  - 2006 and 2011

IEEE 802.15.4-2006 security suites

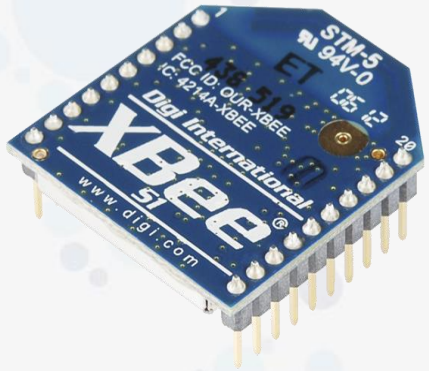
Security Level b_2 b_1 b_0	Security suite	Confidentiality	Integrity
'000'	None	No	No
'001'	MIC-32	No	Yes (M =4)
'010'	MIC-64	No	Yes (M = 8)
'011'	MIC-128	No	Yes (M = 16)
'100'	ENC	Yes	No
'101'	ENC-MIC-32	Yes	Yes (M =4)
'110'	ENC-MIC-64	Yes	Yes (M = 8)
'111'	ENC-MIC-128	Yes	Yes (M = 16)

IEEE 802.15.4-2003 security suites

Security Identifier	Security suite	Confidentiality	Integrity
0x00	None	No	No
0x01	AES-CTR	Yes	No
0x02	AES-CCM-128	Yes	Yes
0x03	AES-CCM-64	Yes	Yes
0x04	AES-CCM-32	Yes	Yes
0x05	AES-CBC-MAC-128	No	Yes
0x06	AES-CBC-MAC-64	No	Yes
0x07	AES-CBC-MAC-32	No	Yes

Deviations for the standard

- **One supplier builds the whole infrastructure**
- **Suppliers design their own firmware**
  - Using SoC solutions
  - Complying with the customer's specification
- **Deviations can stay unnoticed unless…**
  - Availability failures
  - Performance issues
- **Digi XBee S1**
  - 2003 header with 2006 encryption suites
  - Available since 2010 and yet no mention of this anywhere

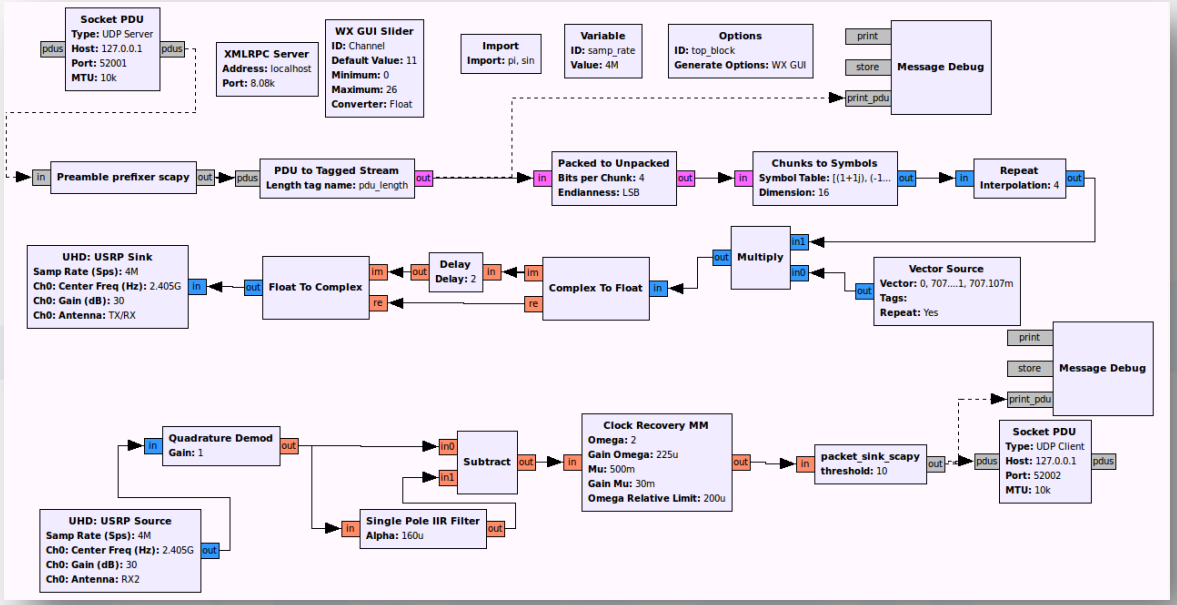


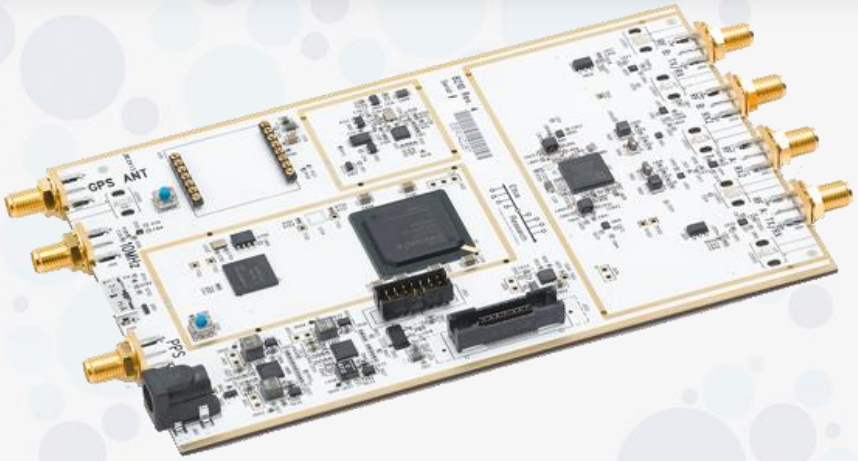
The ARSEN project

- **Advanced Routing between 6LoWPAN and Ethernet Networks**
- **Detecting the configuration of existing 802.15.4 infrastructures**
  - Network topology
  - Data transfer model
  - Security suite
  - Standard revision
  - Standard deviations
- **Handling frame translation between IPv6 and 6LoWPAN**
  - Compression/decompression
  - Fragmentation/defragmentation
  - Support all possible IEEE 802.15.4 configurations

Based on Scapy-radio

```
>>> pckt = Dot15d4FCS() / Dot15d4Data() / ZigbeeNWK()
>>> pckt.show()
###[ 802.15.4 ]###
  fcf_reserved_1= 0
  fcf_panidcompress= False
  fcf_ackreq= False
  fcf_pending= False
  fcf_security= False
  fcf_frametype= Data
  fcf_srcaddrmode= None
  fcf_framever= 0
  fcf_destaddrmode= Short
  fcf_reserved_2= 0
  seqnum= 1
###[ 802.15.4 Data ]###
     dest_panid= 0xffff
     dest_addr= 0xffff
###[ Zigbee Network Layer ]###
        discover_route= 0
        proto_version= 2
        frametype= data
        flags=
        destination= 0x0
        source= 0x0
        radius= 0
        seqnum= 1
>>>
```





https://bitbucket.org/cybertools/scapy-radio

The two main components

- **The IEEE 802.15.4 scanner**
  - Build a database of devices and captured frames
  - The devices that are running on a given channel
  - The devices that are communicating with each other
  - The types of frames that are exchanged between devices
  - The parameters that are used to transmit these frames
- **The 6LoWPAN border router**
  - TUN interface
  - Ethernet omitted (for now)
  - Scapy automaton

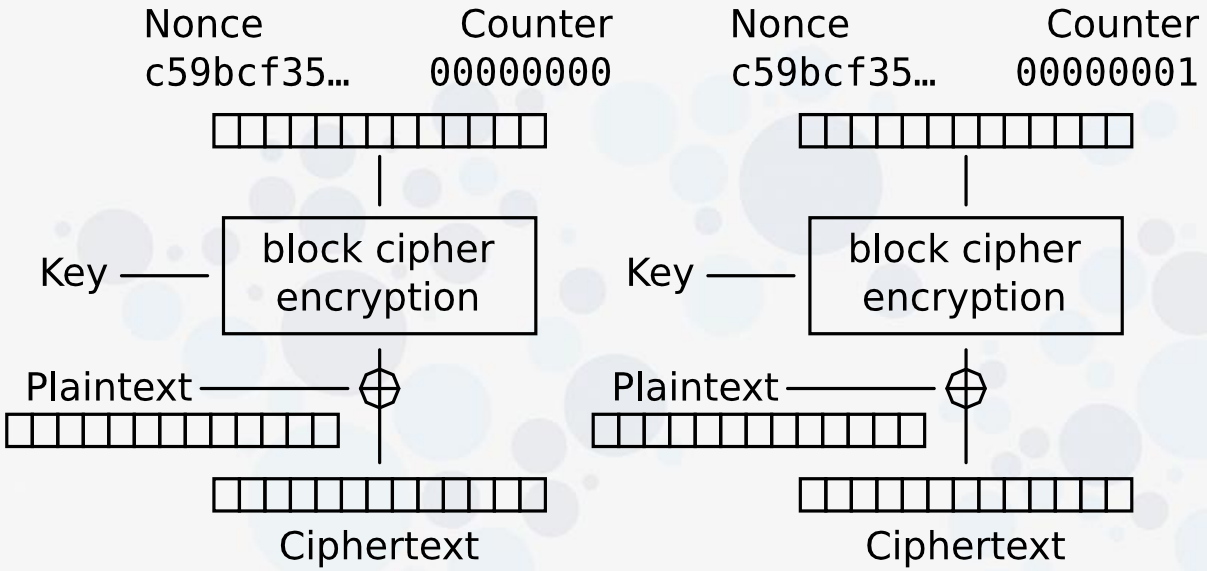
New Scapy layers

- **Dot15d4.py**
  - Several bug fixes
  - Complete 2003 and 2006 support
  - User-provided keystreams support
- **Sixlowpan.py**
  - Uncompressed IPv6 support
  - Complete IP header compression support
  - UDP header compression support
  - Fragmentation and defragmentation support

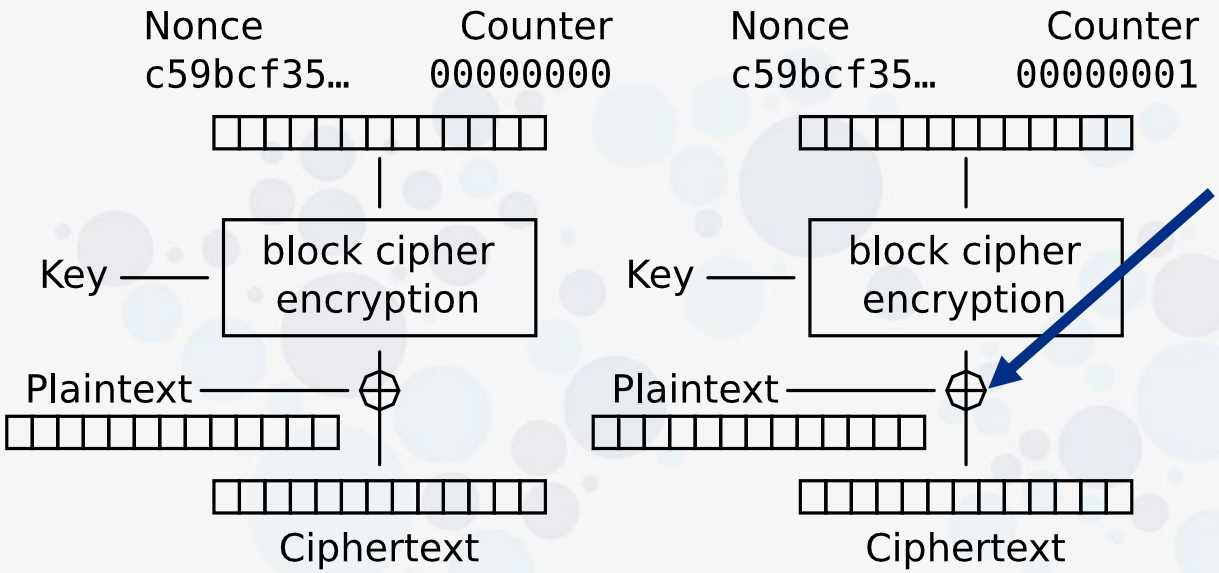
IEEE 802.15.4 known attacks

- **On availability**
  - In theory, the only possible attacks
  - Equivalent to PHY-based jamming attacks
  - Deal with this from a safety point of view (i.e., reboot)
- **On confidentiality**
  - In practice, simplified key management
  - Consequently, same-nonce attacks
- **On integrity**
  - In practice, encryption-only approach and misuse of non-volatile memory
  - Consequently, replay and malleability attacks

AES-CTR (2003) or CCM*-ENC (2006)



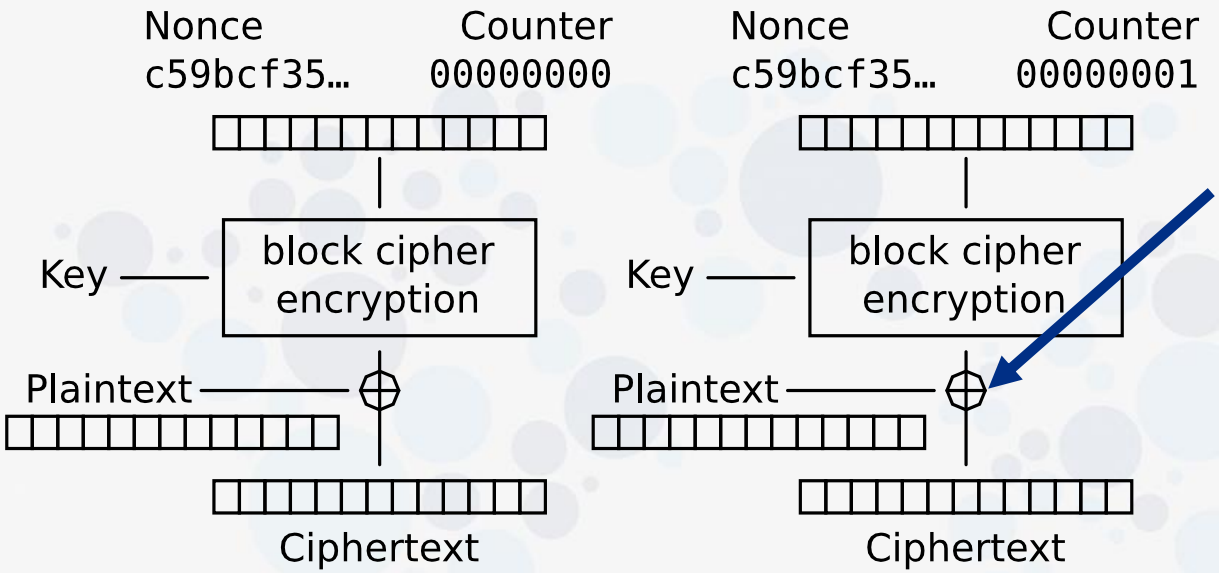
AES-CTR (2003) or CCM*-ENC (2006)



K = F(**Key**, **Nonce**, **AES Counter**)

With K the keystream

AES-CTR (2003) or CCM*-ENC (2006)

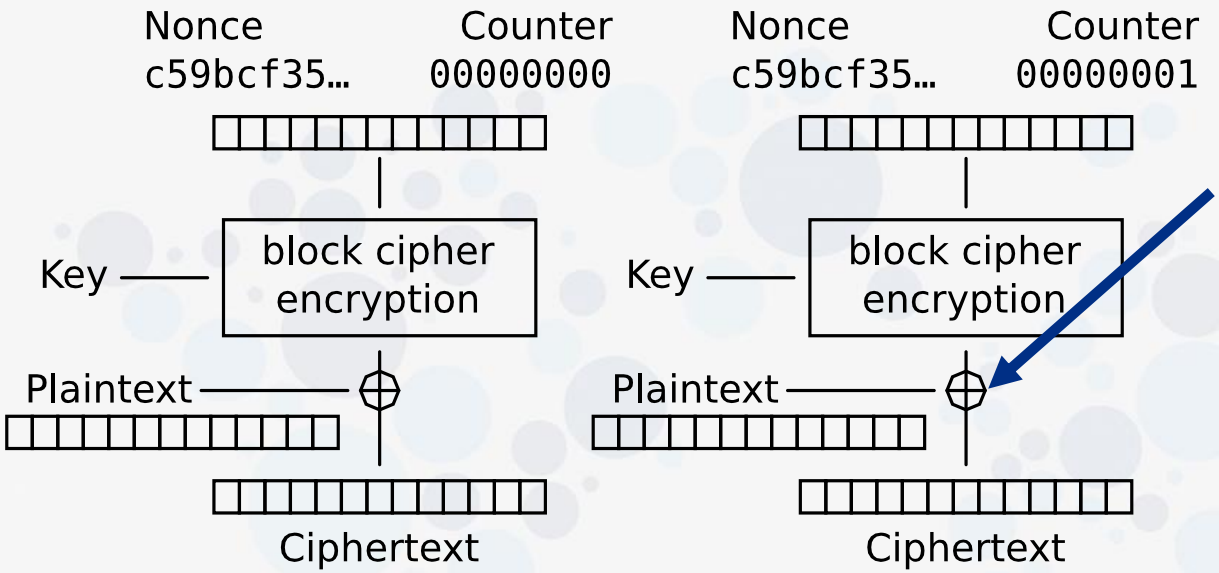


K = F(**Key**, **Nonce**, **AES Counter**)

With K the keystream

Nonce = F(**SrcExtID**, **Frame Counter**)

AES-CTR (2003) or CCM*-ENC (2006)



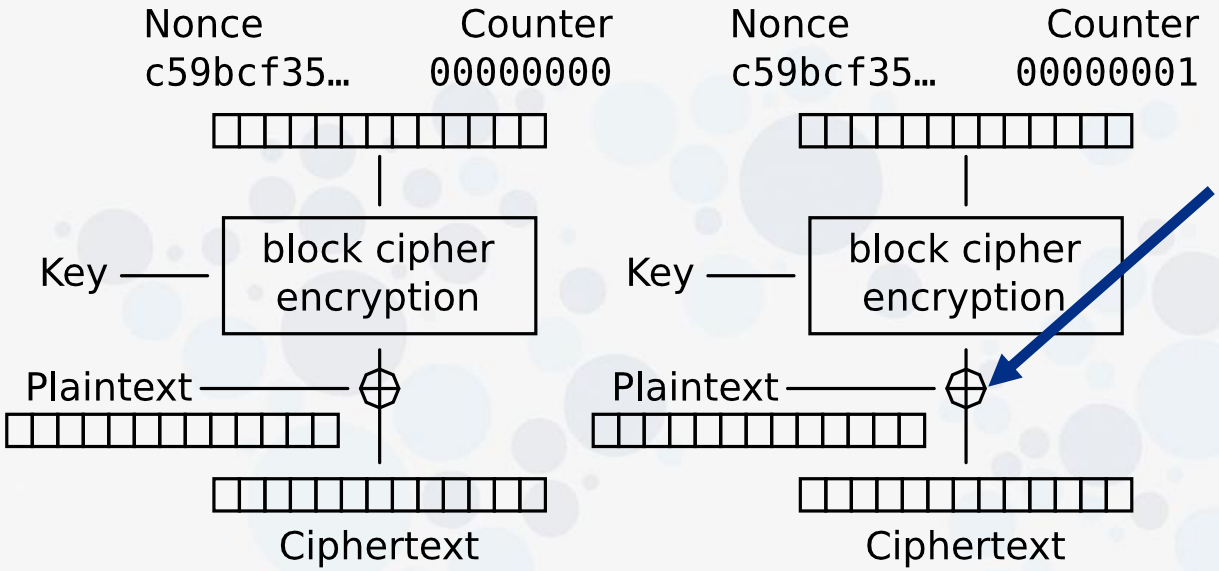
K = F(**Key**, **Nonce**, **AES Counter**)

With K the keystream

Nonce = F(**SrcExtID**, **Frame Counter**)

C⊗C' = (P⊗K)⊗(P'⊗K)= **P⊗P'**

AES-CTR (2003) or CCM*-ENC (2006)



K = F(**Key**, **Nonce**, **AES Counter**)

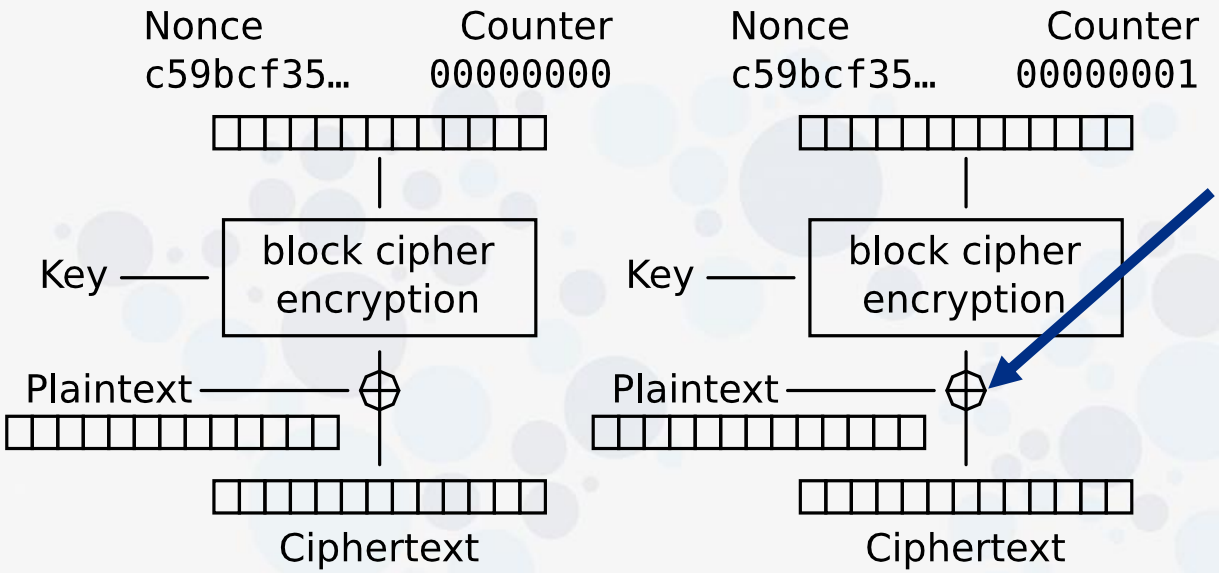
With K the keystream

Nonce = F(**SrcExtID**, **Frame Counter**)

C⊗C' = (P⊗K)⊗(P'⊗K)= **P⊗P'**

- **Same-nonce attacks**
  - If one captured frame is known or guessable
  - Or statistical analysis on a large number of captured frames

AES-CTR (2003) or CCM*-ENC (2006)



K = F(**Key**, **Nonce**, **AES Counter**)

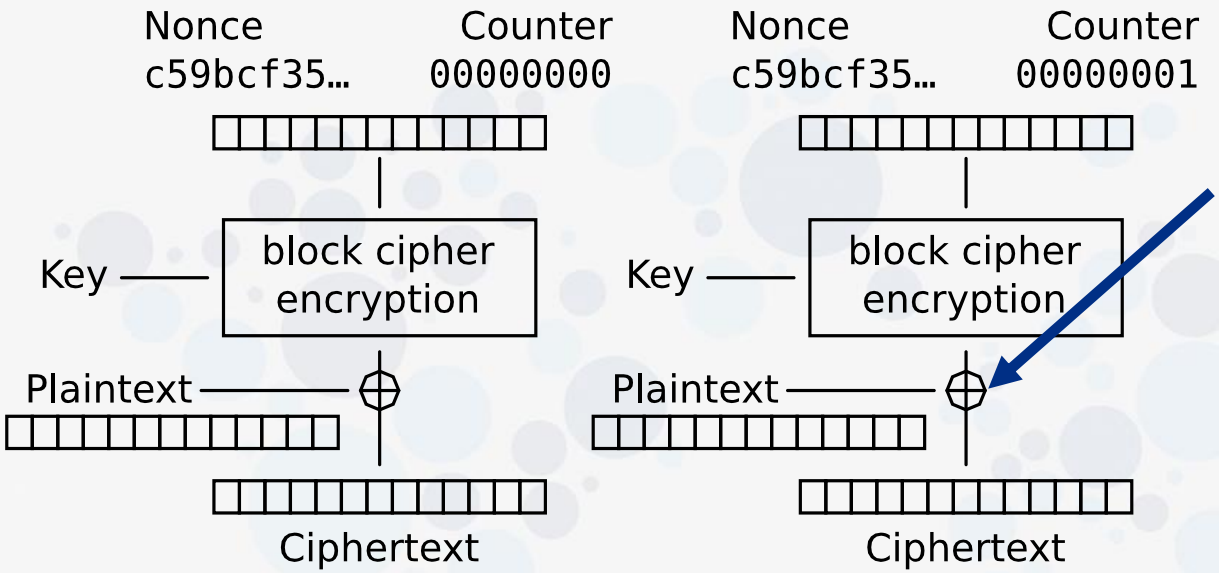
With K the keystream

Nonce = F(**SrcExtID**, **Frame Counter**)

C⊗C' = (P⊗K)⊗(P'⊗K)= **P⊗P'**

- **Replay attacks**
  - Frame counters not being checked
  - Frame counters not being stored in non-volatile memory

AES-CTR (2003) or CCM*-ENC (2006)



K = F(**Key**, **Nonce**, **AES Counter**)

With K the keystream

Nonce = F(**SrcExtID**, **Frame Counter**)

C⊗C' = (P⊗K)⊗(P'⊗K)= **P⊗P'**

- **Malleability attacks (useful when no physical access)**
  - Keystreams provided by same-nonce attacks (with a simple XOR)
  - Frame counters allowed by replay attacks

Application on a metering infrastructure

- **Monitoring of a water distribution system**
  - Wireless sensor network
  - Focus on two particular reachable sensors







Information gathering

- **Using the ARSEN scanner**
  - Channel 18 is used for transmission
  - Sensors only communicate with the PAN_Coord
  - PAN_Coord is only transmitting beacon frames
  - Frame version: IEEE 802.15.4-2006 standard
  - Security functions are used: AES-CTR mode
  - Short_Addr are used, we will need Long_Addr

Transmitter0:
beacon_enabled=0x1
pan_coord=0x1
coord=0x1
gts=0x0
panid=0xabba
short_addr=0xde00

Transmitter1:
short_addr=0xde02
panid=0xabba
Destination0:
security_enabled=0x1
frame_version=0x1L
short_addr=0xde00
coord=0x1
command=0x0
panid=0xabba
data=0x5
pan_coord=0x1

Transmitter2:
short_addr=0xde01
panid=0xabba
Destination0:
security_enabled=0x1
frame_version=0x1L
short_addr=0xde00
coord=0x1
command=0x0
panid=0xabba
data=0x4
pan_coord=0x1

Information gathering

- **We need long addresses**
  - They are used to compute the nonce
  - They are sent during association
- **How to force re-association**
  - Sensors are tracking beacons
  - Use Scapy-radio with the new Dot15d4 layer
  - Flood the channel to disrupt the PAN
  - The sensors cannot track beacon frames
  - The sensors go into synchronization-loss state
  - They then try to re-associate

```
Transmitter0 :
beacon_enabled=0x1
pan_coord=0x1
coord=0x1
long_addr=0x158d000053da9d
gts=0x0
panid=0xabba
short_addr=0xde00
Destination0:
                frame_version=0x0L
                short_addr=0xde01
                command=0x1
                panid=0xabba
                data=0x0
                long_addr=0x158d00005405a6
Destination1:
                frame_version=0x0L
                short_addr=0xde02
                command=0x1
                panid=0xabba
                data=0x0
                long_addr=0x158d0000540591
```

The association procedure

- **Analysis of captured association frames**
  - No secure function are used during association
  - No higher protocol are used for authentication
  - Channels 11 to 26 are scanned (with beacon requests)
- **Adding a fake sensor to the network**
  - No specific actions are required
  - Any long address is accepted by the PAN coordinator
  - No need to spoof an actual sensor (unless we want to replay frames)
  - We will not be able to send encrypted frames

Outgoing frame counters

- **Expected behavior: reboot of sensors when loss of synchronization lasts for a determined amount of time**
- **How to force the reboot of sensors**
  - Continuously flood the channel of the PAN coordinator (18)
  - Synchronization is thus lost permanently for sensors
  - Sensors look up for a PAN coordinator on all channels (11 to 26)
  - If beacon requests stop for a moment, then sensors may have rebooted
  - Stop flooding, let re-associations happen and observe the frame counters

  → If they are not stored in non-volatile memory, they will be reset on reboot

Incoming frame counters

- **Similar expected behavior for the PAN coordinator**
- **How to force the reboot of the PAN coordinator**
  - Create a fake PAN coordinator on a channel below 18
  - Force re-association of sensors (to our fake PAN coordinator)
  - If beacons stop for a moment, then the PAN coordinator may have rebooted
  - Wait for beacons to come back (i.e., the PAN coordinator is up gain)
  - Associate a fake sensor and replay previously captured frames
  - If the beacons never stop again, replayed frames have thus been accepted

  → The counters have been reset (i.e., not stored in non-volatile memory)

Forging encrypted frames

- **We can reset outgoing frames counters**

  → We can thus conduct same-nonce attacks

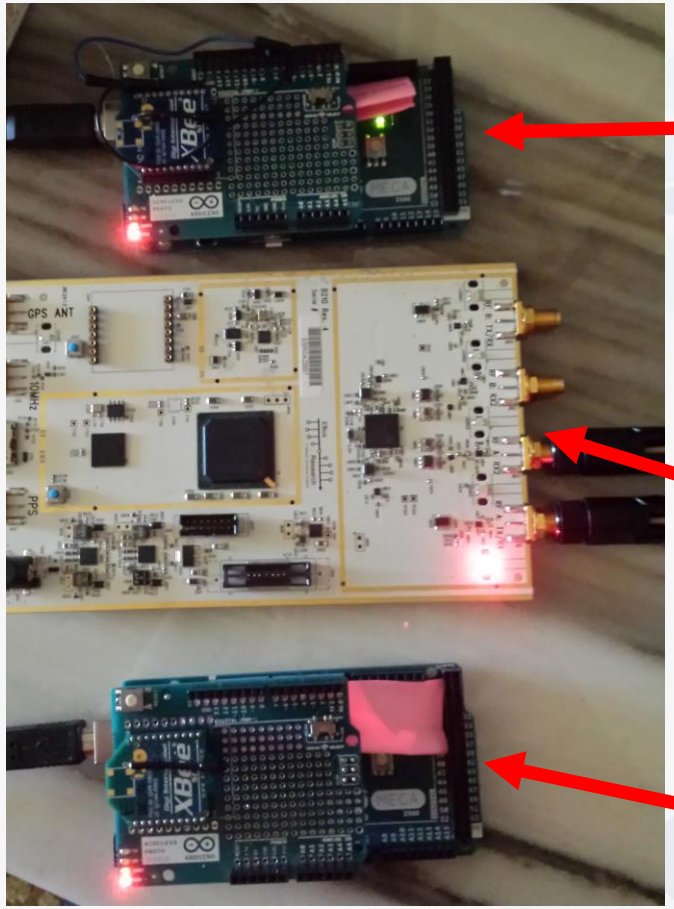
- **We can reset incoming frames counters**

  →We can thus conduct replay attacks

- **Therefore, we can conduct malleability attacks**
  - Create a set of valid keystreams with their corresponding frame counters
  - Provide this set to the new Dot15d4 Scapy layer

- **Finally, set up the ARSEN border router and start auditing higher-layer protocols and their services**

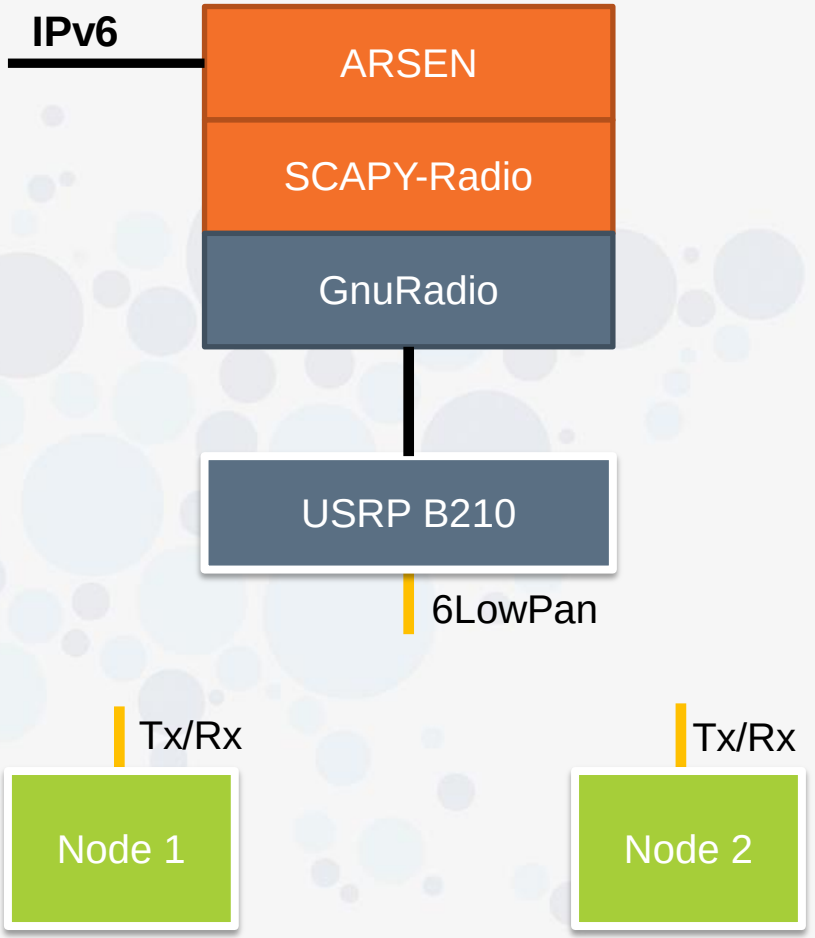
Demonstration bench



Node 1 with XBee S1

USRP B210 used by the ARSEN tools

Node 2 with Xbee S1



Demonstration bench

```
pentest-penhost{~/scapy-radio-tools}: ping6 fe80::0213:a200:40dc:0580 -I tun0 -i 2 -c 10
PING fe80::0213:a200:40dc:0580(fe80::213:a200:40dc:580) from fe80::213:a200:4081:b470 tun0: 56 data bytes
64 bytes from fe80::213:a200:40dc:580: icmp_seq=1 ttl=255 time=753 ms
64 bytes from fe80::213:a200:40dc:580: icmp_seq=2 ttl=255 time=775 ms
64 bytes from fe80::213:a200:40dc:580: icmp_seq=4 ttl=255 time=847 ms
64 bytes from fe80::213:a200:40dc:580: icmp_seq=5 ttl=255 time=829 ms
64 bytes from fe80::213:a200:40dc:580: icmp_seq=6 ttl=255 time=752 ms
64 bytes from fe80::213:a200:40dc:580: icmp_seq=7 ttl=255 time=775 ms
64 bytes from fe80::213:a200:40dc:580: icmp_seq=8 ttl=255 time=796 ms
64 bytes from fe80::213:a200:40dc:580: icmp_seq=9 ttl=255 time=873 ms
64 bytes from fe80::213:a200:40dc:580: icmp_seq=10 ttl=255 time=756 ms

--- fe80::0213:a200:40dc:0580 ping statistics ---
10 packets transmitted, 9 received, 10% packet loss, time 18010ms
rtt min/avg/max/mdev = 752.978/795.600/873.929/41.963 ms
pentest-penhost{~/scapy-radio-tools}: nmap -p 80,443,22 fe80::0213:a200:40dc:0580%tun0 -T 2 -6

Starting Nmap 6.46 ( http://nmap.org ) at 2016-08-07 20:39 CEST
Nmap scan report for fe80::213:a200:40dc:580
Host is up (1.3s latency).
PORT    STATE SERVICE
22/tcp  open  ssh
80/tcp  open  http
443/tcp open  https

Nmap done: 1 IP address (1 host up) scanned in 8.94 seconds
pentest-penhost{~/scapy-radio-tools}: telnet fe80::0213:a200:40dc:0580%tun0 80
Trying fe80::213:a200:40dc:580%tun0...
Connected to fe80::0213:a200:40dc:0580%tun0.
Escape character is '^]'.
Hello World
```

Thank you for your attention

https://bitbucket.org/cybertools/scapy-radio