

# SweynTooth: Unleashing Mayhem over Bluetooth Low Energy

Matheus Eduardo Garbelini<sup>1</sup>, Chundong Wang<sup>2</sup>, Sudipta Chattopadhyay<sup>1</sup>,  
Sun Sumei<sup>3</sup>, Ernest Kurniawan<sup>3</sup>

<sup>1</sup> Singapore University of Technology and Design (SUTD)

<sup>2</sup> ShanghaiTech University. Work partly done when C. Wang worked at SUTD

<sup>3</sup> Institute for Infocomm Research, A\*Star



SINGAPORE UNIVERSITY OF  
TECHNOLOGY AND DESIGN



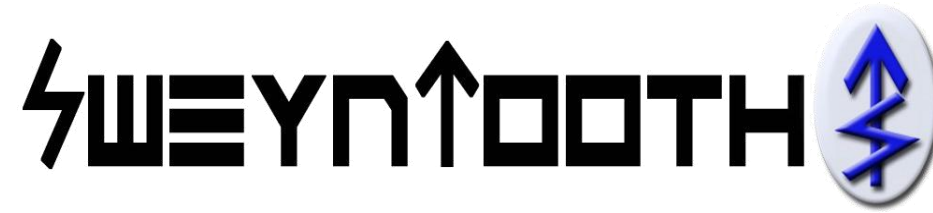
Institute for  
Infocomm Research

USENIX Annual Technical Conference 2020, July 15-17  
Track 2, The One on the Edge

Partially sponsored by Keysight Technologies



# Why the Mayhem?



*A family of over dozen new vulnerabilities in Bluetooth Low Energy (BLE) implementations*  
Named after *Sweyn Forkbeard* who revolted against his father *King Harald Bluetooth*.

## Affected SoC Vendors (not exhaustive)



## Many IoTs affected



(a) FitBit Inspire



(b) Eve Energy



(c) August Smart Lock



(d) CubiTag



(e) eGeeTouch

## Open Source BLE Stack (not exhaustive)



Arm Mbed Cordio

# A look into Bluetooth flavours - Past Vulnerabilities

Is everything well tested?

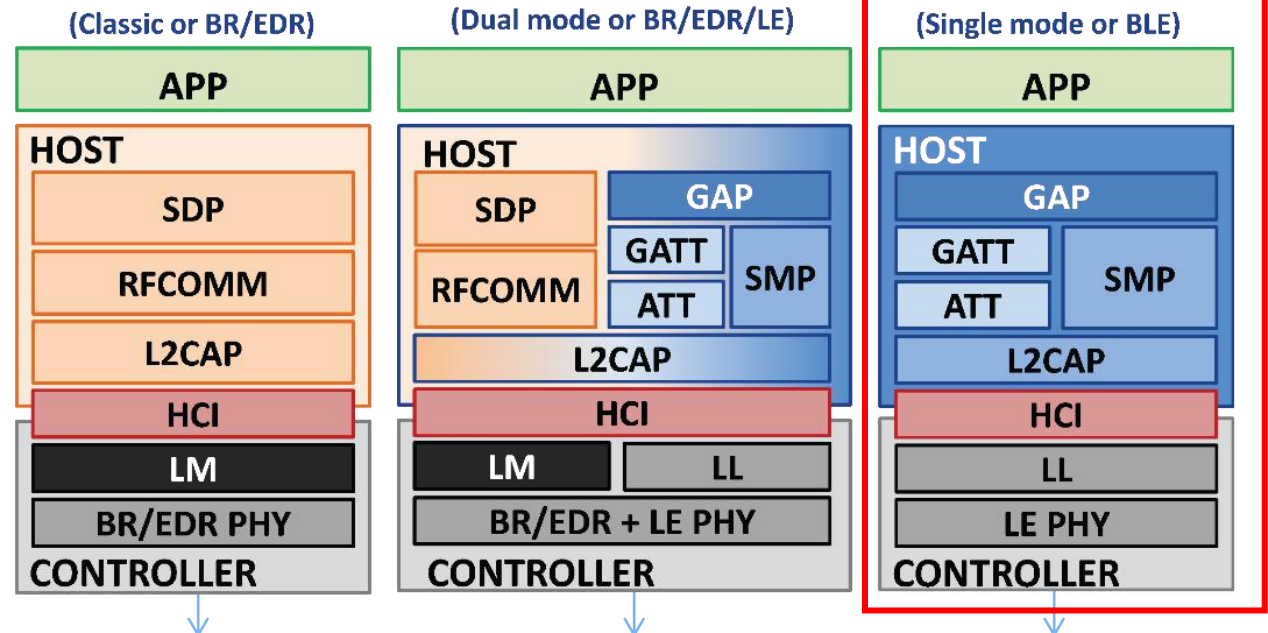
## Latest Attacks

- [2017] BlueBorne (Classic)
- [2018] BleedingBit (BLE)
- [2019] Invalid Curve Attack (Classic/BLE)
- [2019] Knob (Classic)
- [2020] Bias (Classic)

Affected stack  
(Classic)  
(BLE)  
(Classic/BLE)  
(Classic)  
(Classic)

More complexity  
More vulnerabilities!

Our Target



# Bluetooth Low Energy Overview

Can we test BLE security ourselves with off the shelf hardware?

Device roles – Central vs peripheral

Smartphone



Advertises  
Peripheral

Our target

Smart  
Watch



Connects to  
Central



Standard Testing Equipment



Ellisys Bluetooth  
Explorer (Over \$10k)

Can we avoid  
this setup?

# Bluetooth Low Energy Overview

Can we test BLE security ourselves with off the shelf hardware?

Device roles – Central vs peripheral

Smartphone



Advertises  
**Peripheral**

Our target

Connects to

**Central**

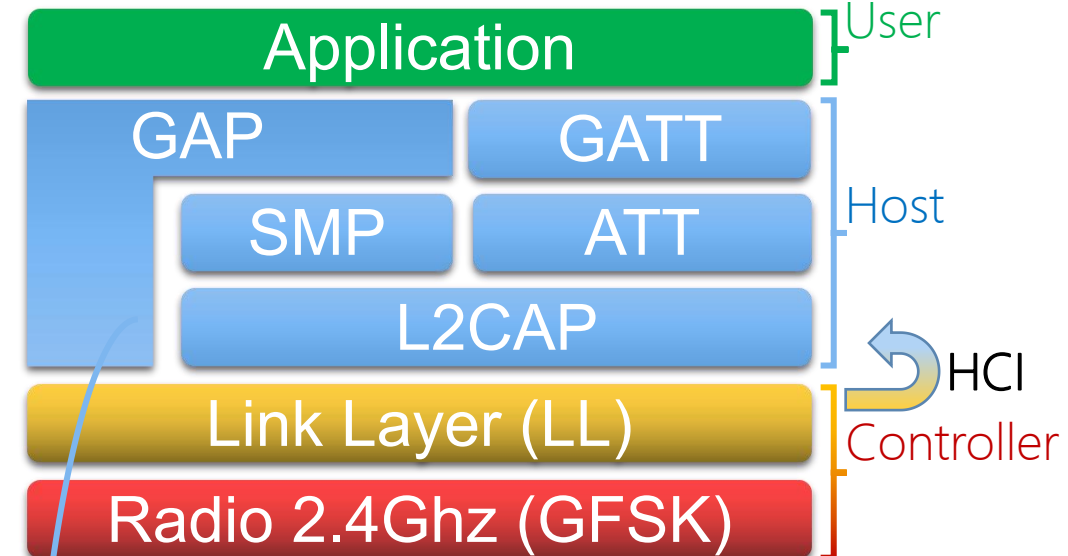
Smart Watch



BLE Frame



BLE Stack





# Bluetooth Low Energy Overview

Can we test BLE security ourselves with off the shelf hardware?

Device roles – Central vs peripheral

Smartphone



Connects to  
**Central**

Advertises  
**Peripheral**

Our target

Smart  
Watch

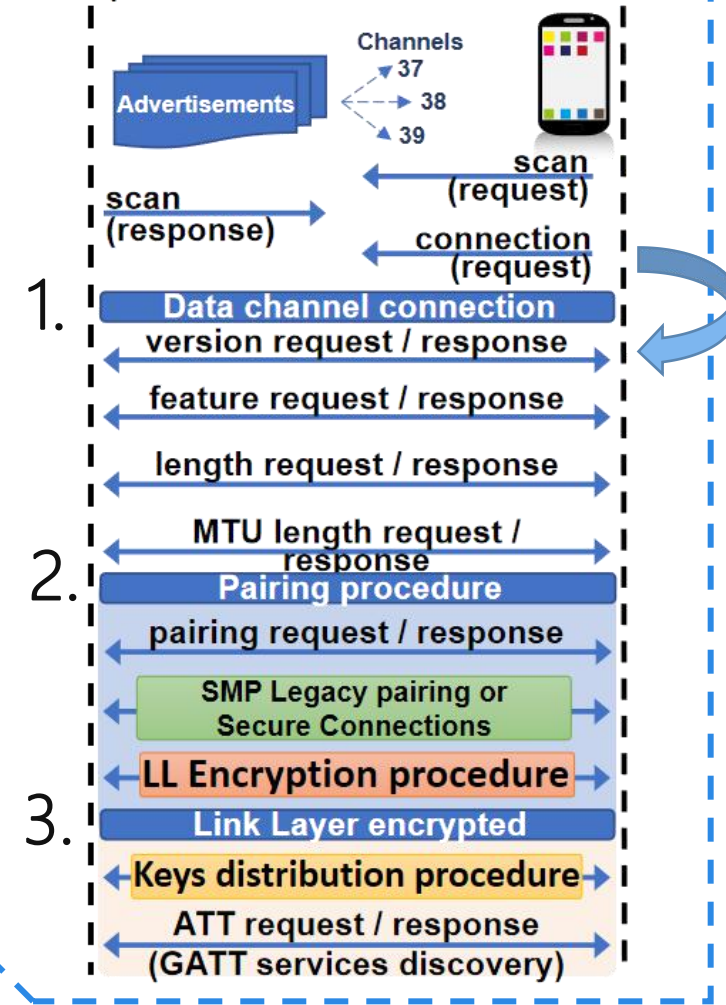


1. Peripheral switches from advertisement channels to data channels;
2. Pairing procedure is performed according to device capabilities;
3. Link Layer encryption (managed only by the controller).

## Main BLE Exchanges

Peripheral/Slave

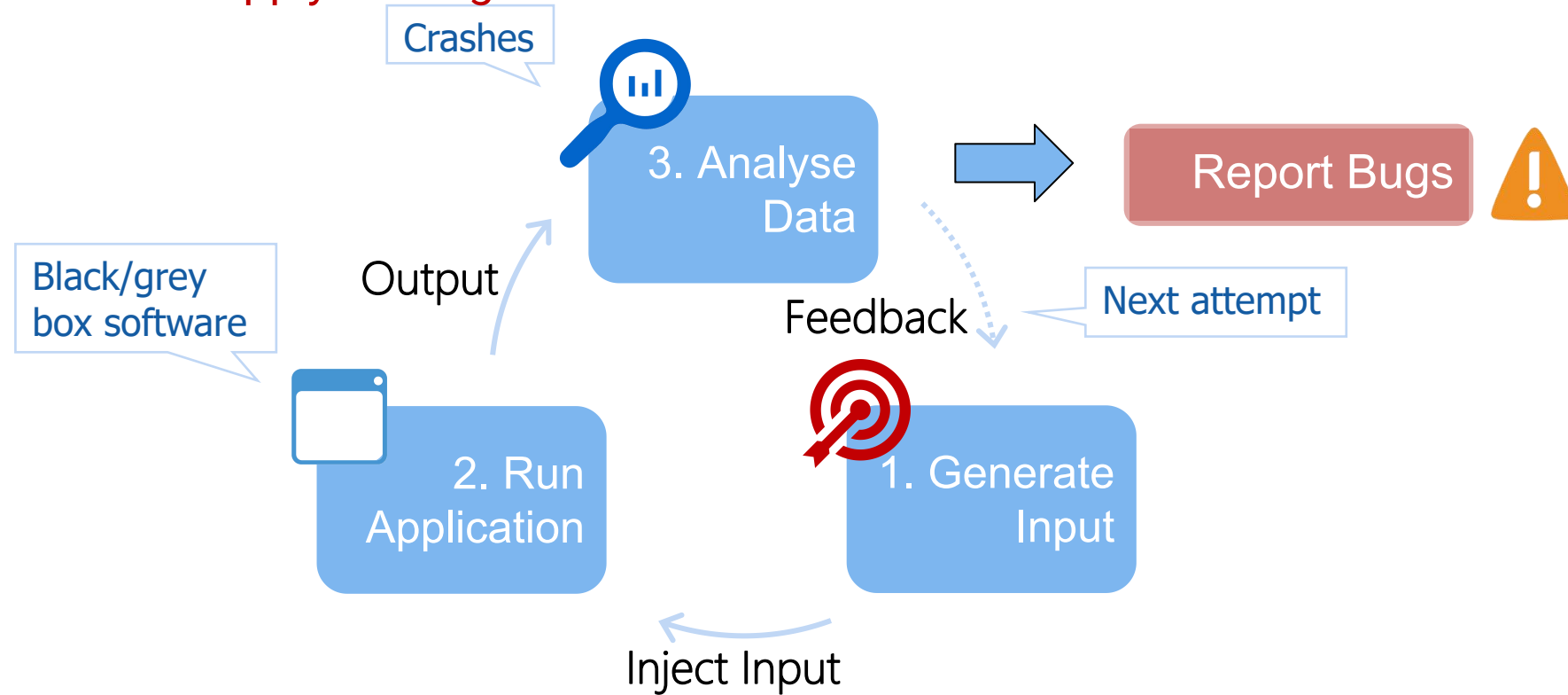
Central/Master



From adv. channel to data channel

# Testing Security by Fuzzing

Is it possible to apply fuzzing to lower-level over the air communication?

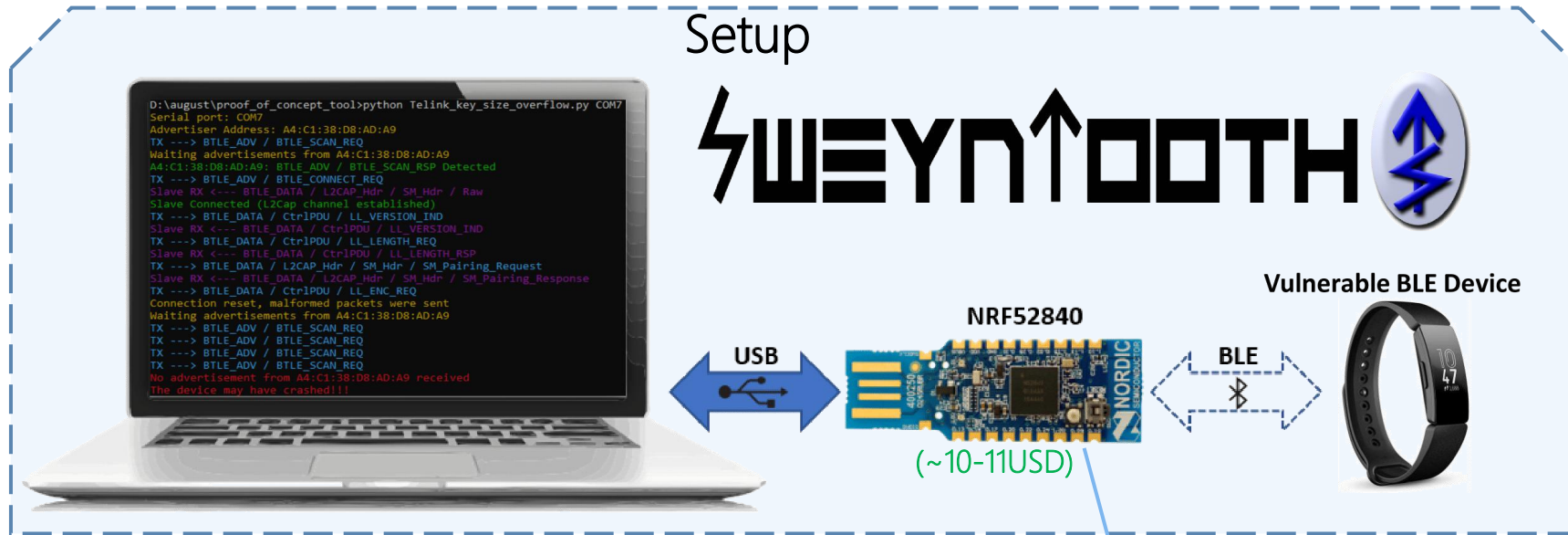


## Challenges:

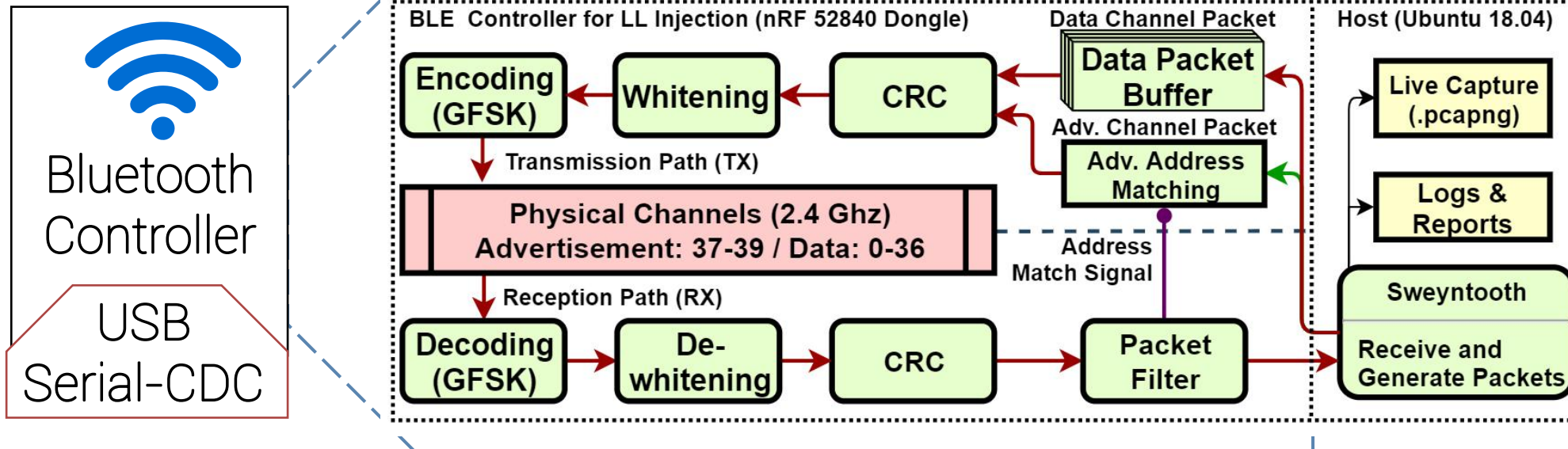
1. Full control over BLE Link Layer (Including manipulation of the **connection procedure**)
2. What feedback metric to use? Most BLE stack implementation is closed source.
3. BLE is a heavily stateful protocol, simply mutating the input is not enough.
4. How to detect crashes or anomalies when fuzzing **over the air**?

# Introducing a non-compliant controller implementation!

Setup



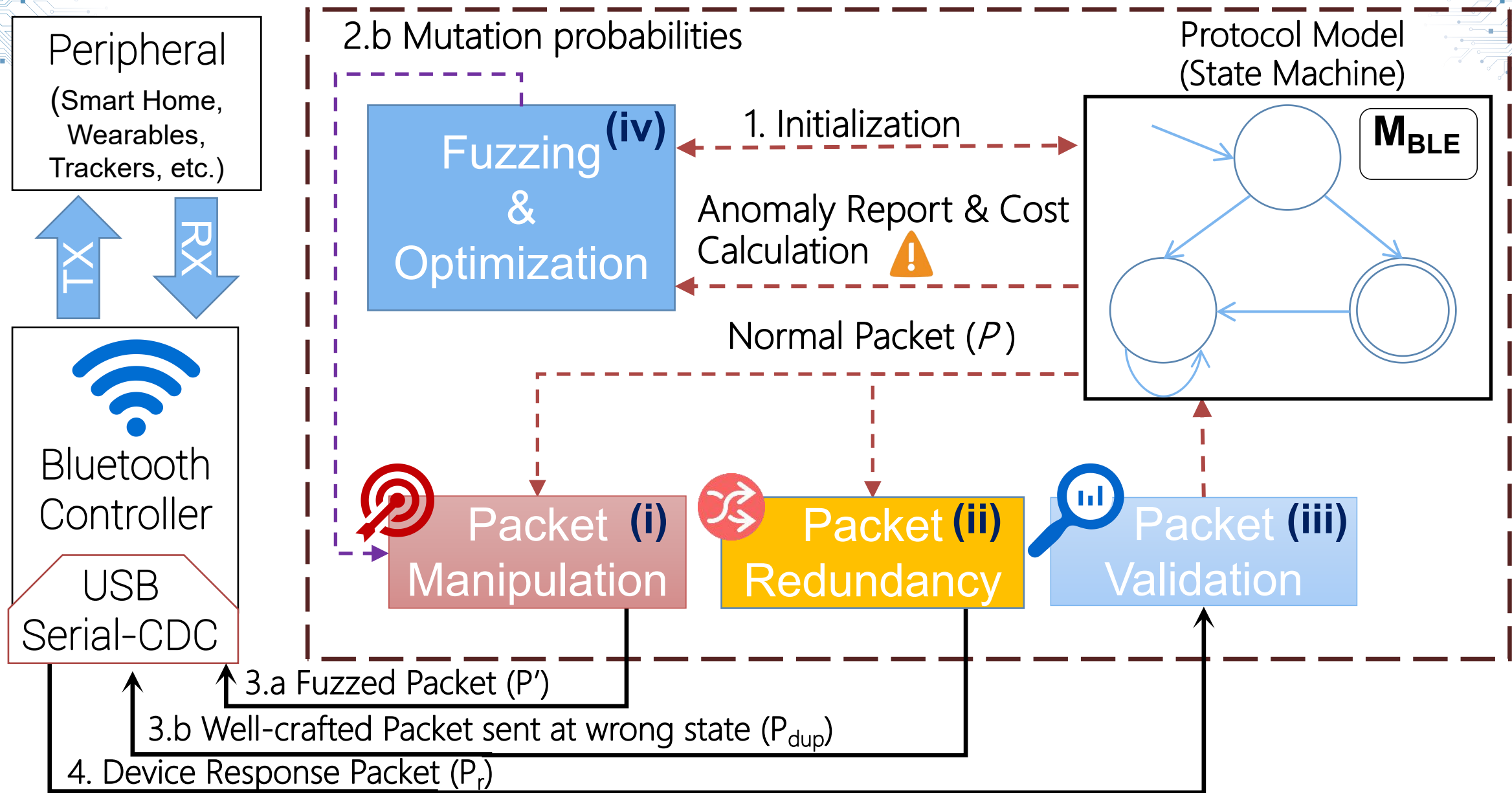
Internal Design



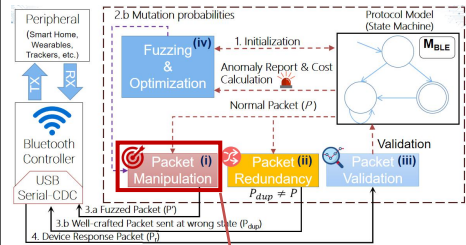
**CAUTION**  
No HCI allowed here!




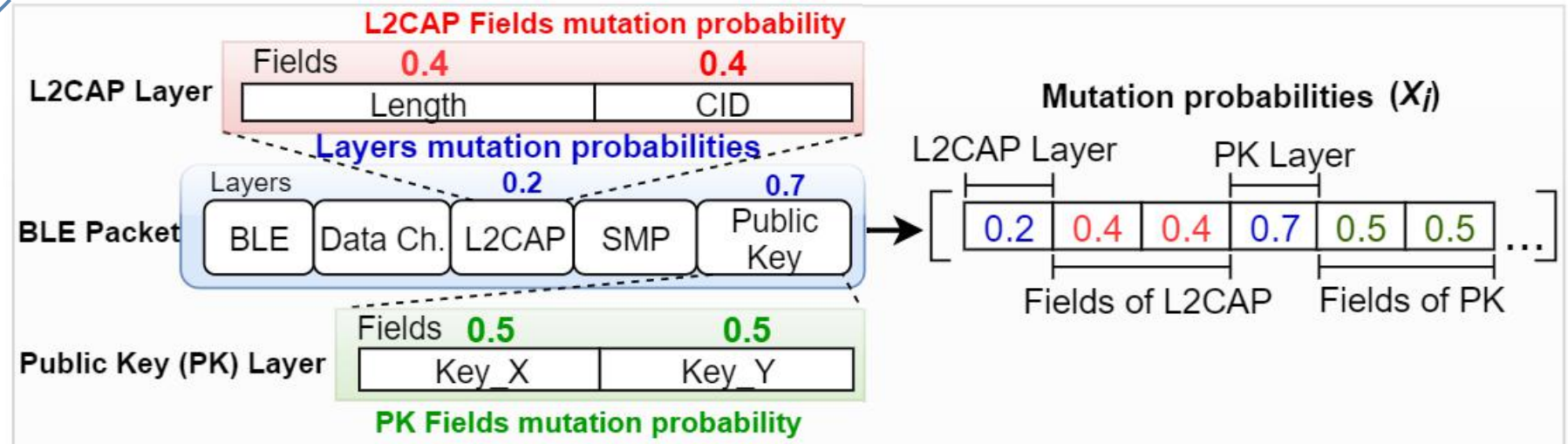
# Fuzzer Architecture Overview



# Fuzzing BLE Layers - Fields mutation

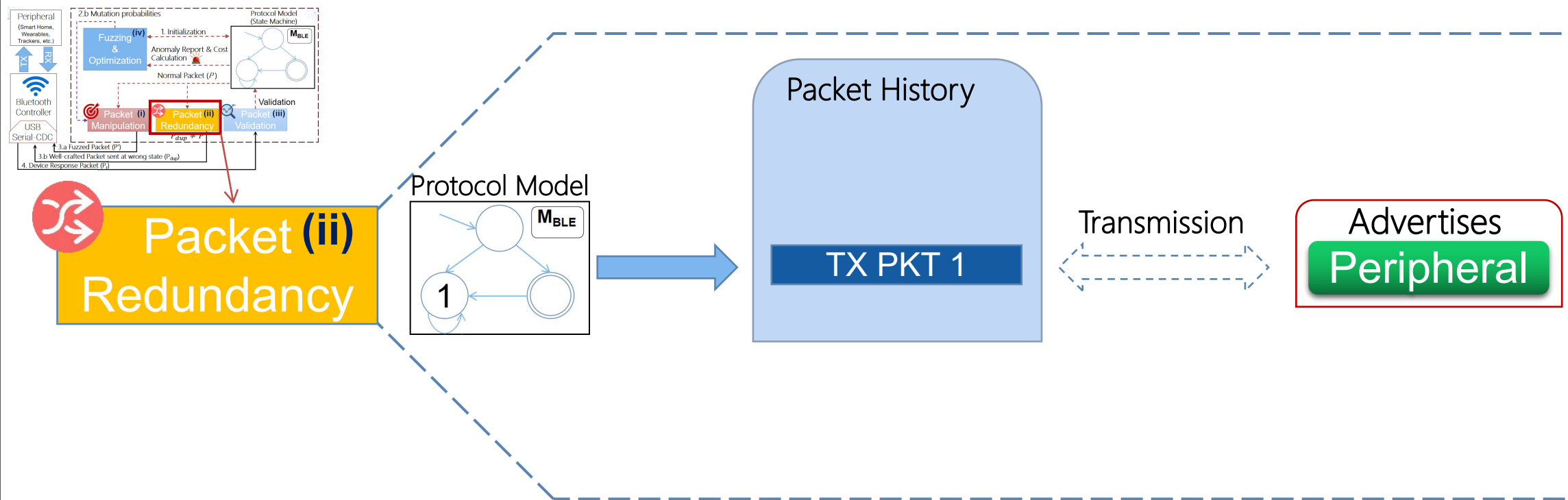


 **Packet (i)**  
**Manipulation**



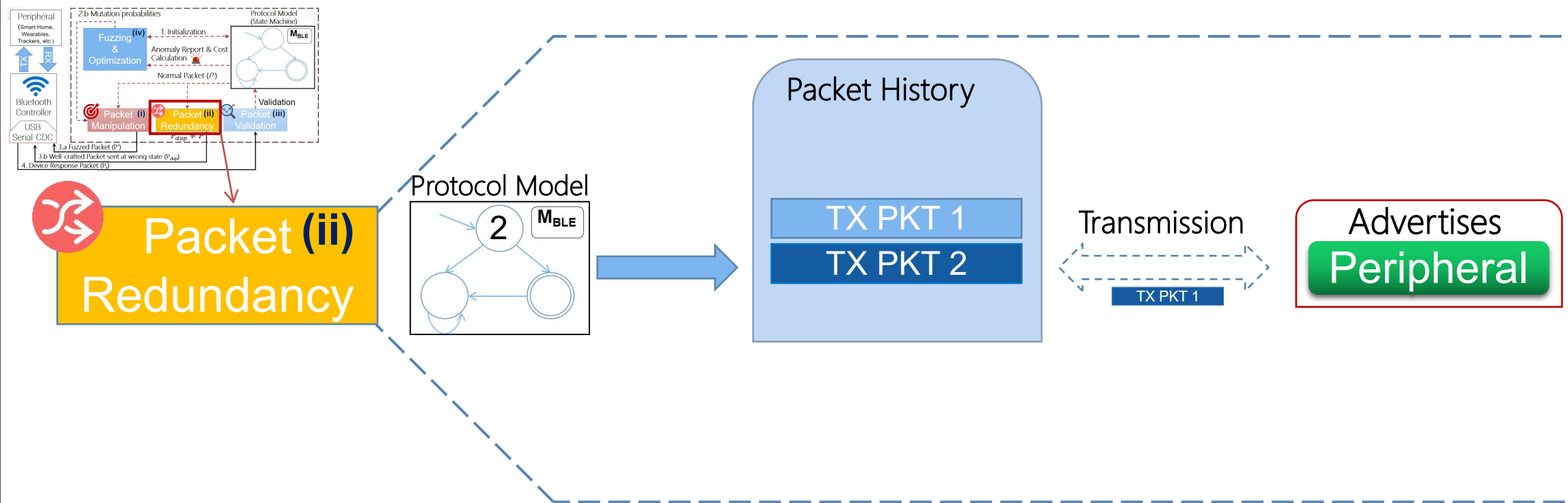


# Fuzzing BLE Layers - Out of order sequences



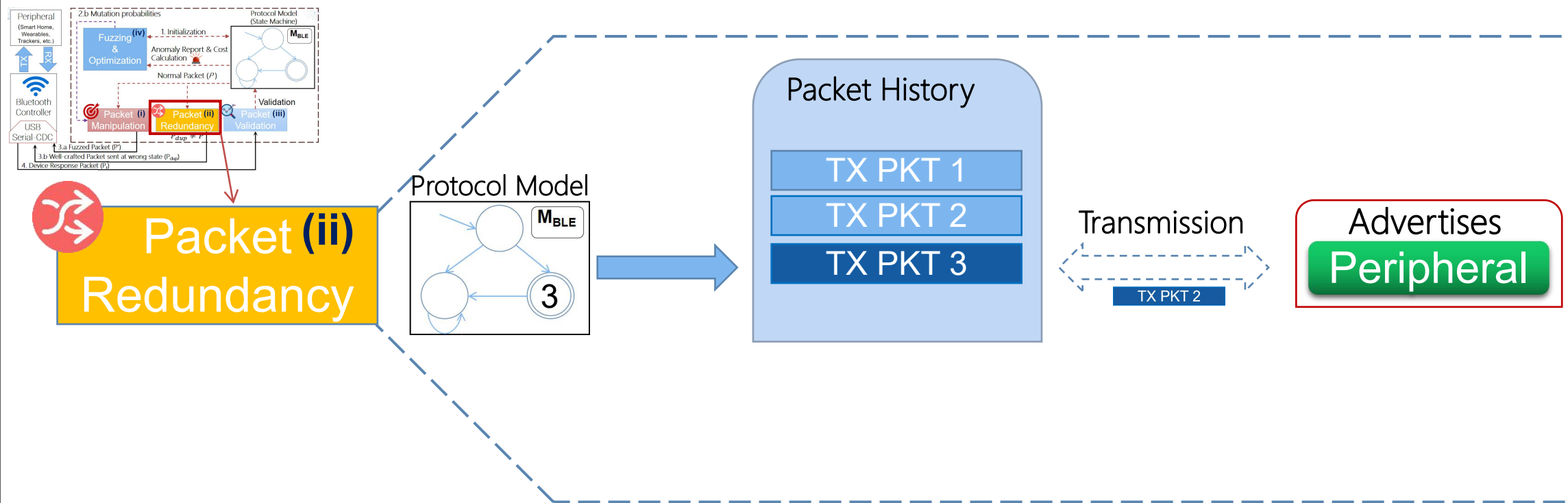


# Fuzzing BLE Layers - Out of order sequences





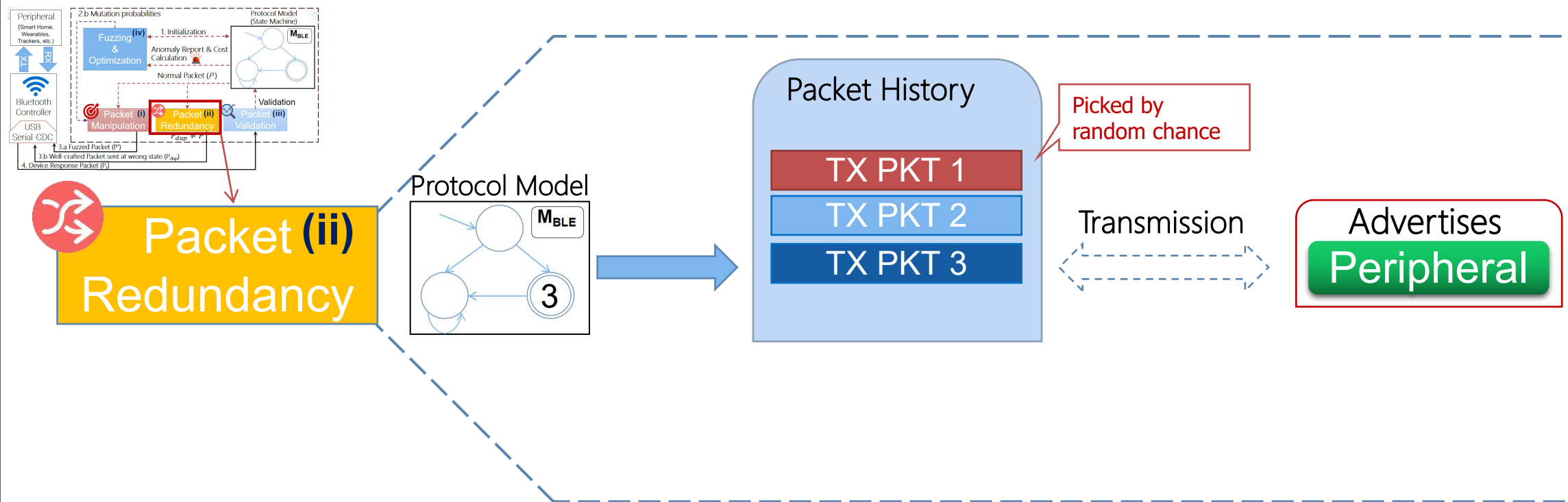
# Fuzzing BLE Layers - Out of order sequences





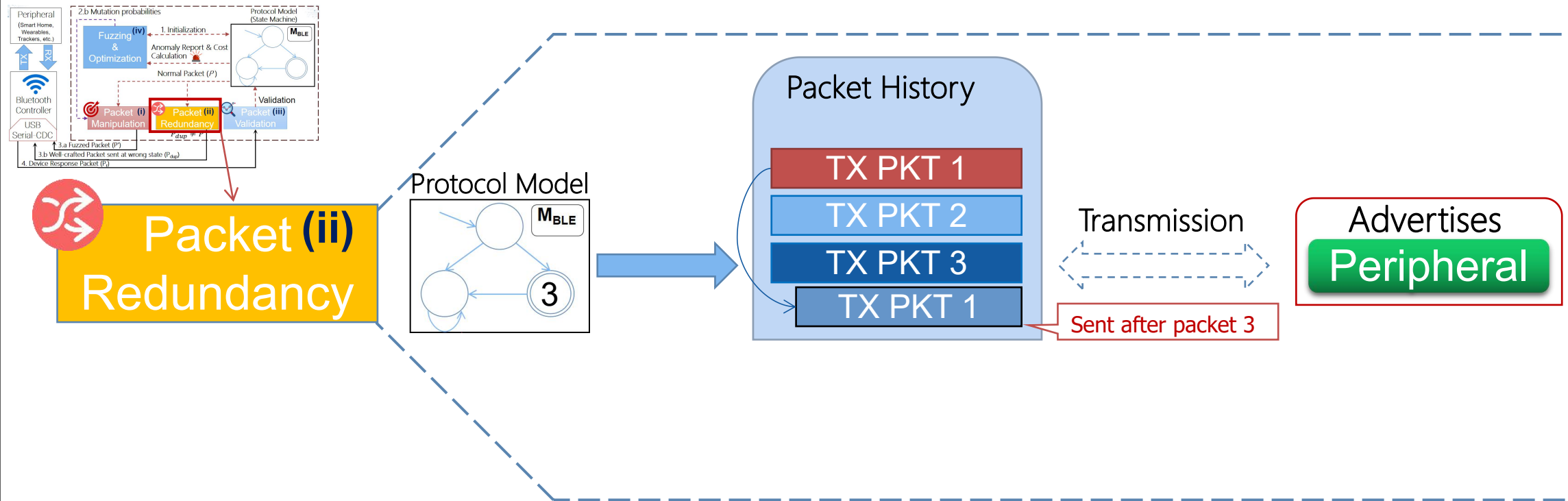


# Fuzzing BLE Layers - Out of order sequences



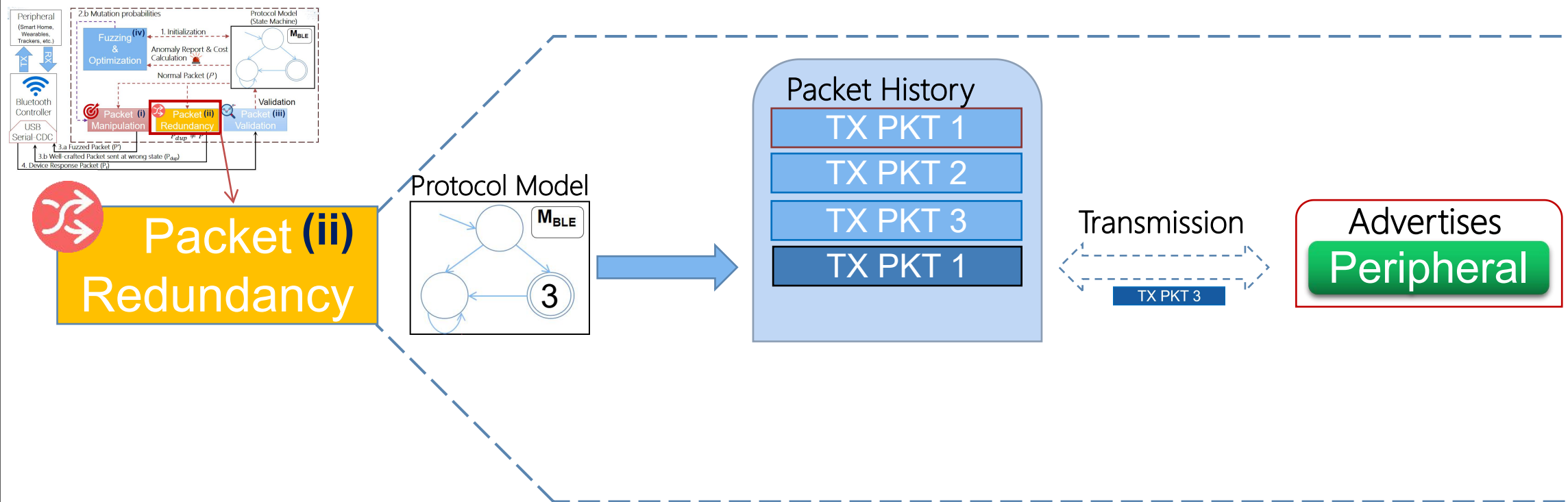


# Fuzzing BLE Layers - Out of order sequences

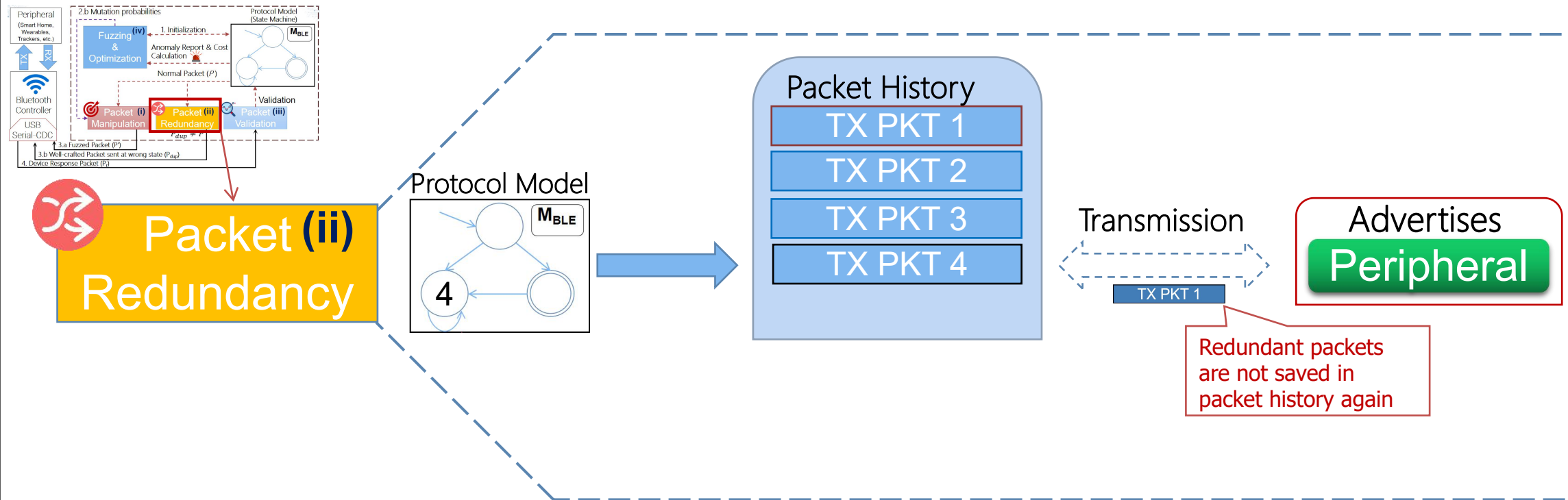




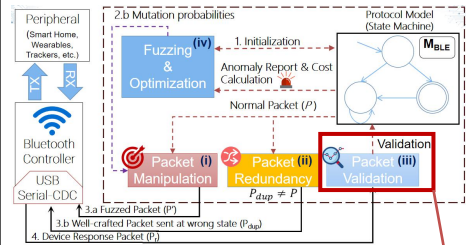
# Fuzzing BLE Layers - Out of order sequences



# Fuzzing BLE Layers - Out of order sequences

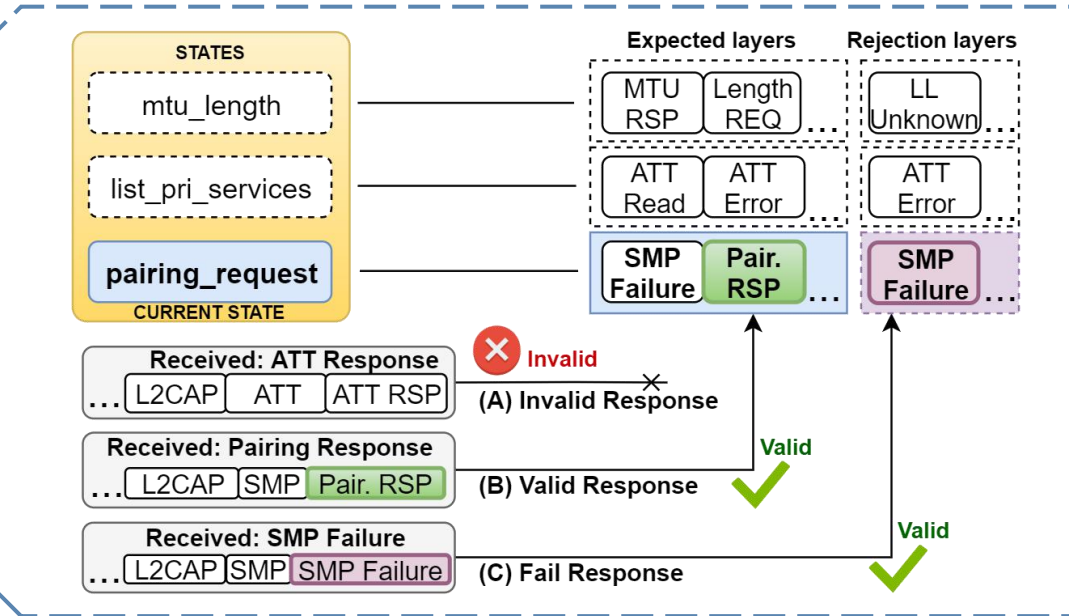


# Validation Strategy - Exemplified



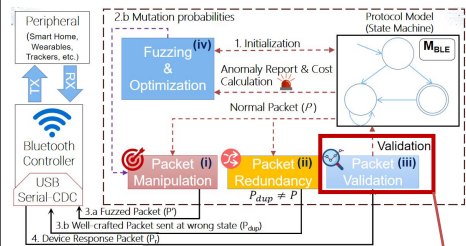
## Packet Validation

### Valid and invalid responses



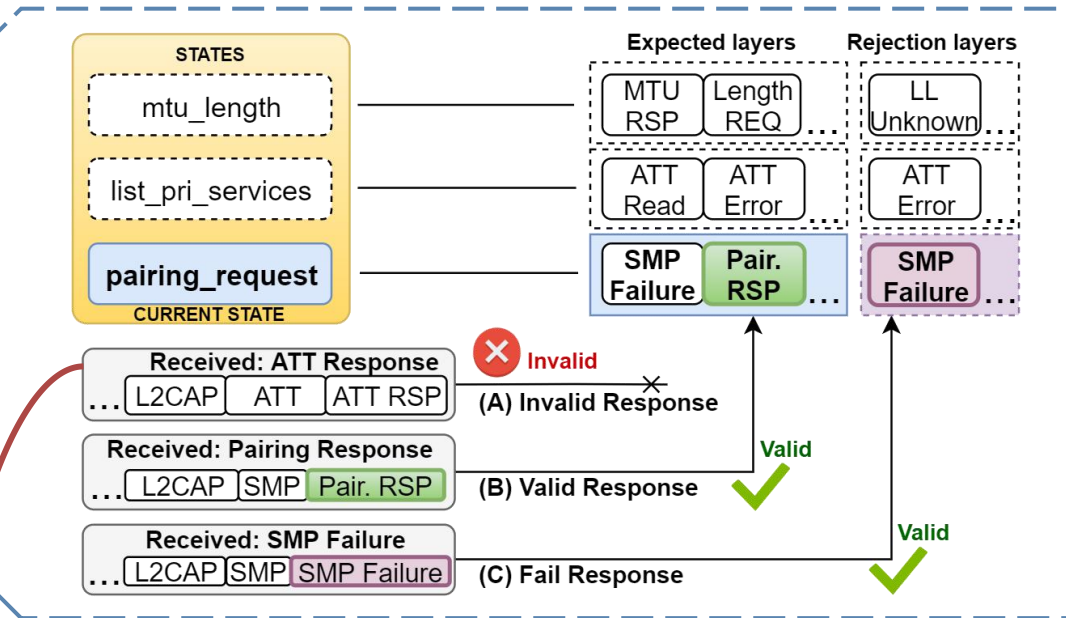


# Validation Strategy - Exemplified



## Packet Validation

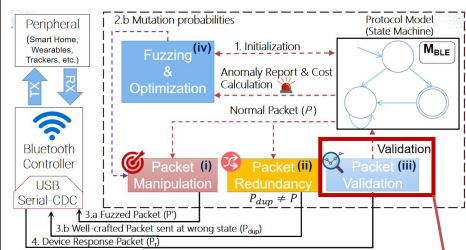
### Valid and invalid responses



373	8.236579	Slave_0x9a328370	Master_0x9a328370	LE LL	Empty PDU
374	8.247405	Master_0x9a328370	Slave_0x9a328370	SMP	Sent Pairing Request: AuthReq: Bonding   Initiator Key(s): LTK, IRK, CSRK   Responder Key(s): LTK, IRK, CSRK
375	8.256444	Slave_0x9a328370	Master_0x9a328370	LE LL	Empty PDU
376	8.276971	Slave_0x9a328370	Master_0x9a328370	ATT	Rcvd Error Response - Attribute Not Found, Handle: 0x002e (Device Information: Unknown)
377	8.316158	Slave_0x9a328370	Master_0x9a328370	LE LL	Empty PDU
378	8.336147	Slave_0x9a328370	Master_0x9a328370	LE LL	Empty PDU
379	8.356301	Slave_0x9a328370	Master_0x9a328370	LE LL	Empty PDU
380	8.377797	Slave_0x9a328370	Master_0x9a328370	LE LL	Empty PDU
381	8.396465	Slave_0x9a328370	Master_0x9a328370	LE LL	Empty PDU
382	8.416231	Slave_0x9a328370	Master_0x9a328370	LE LL	Empty PDU
383	8.436107	Slave_0x9a328370	Master_0x9a328370	LE LL	Empty PDU
384	8.456092	Slave_0x9a328370	Master_0x9a328370	LE LL	Empty PDU

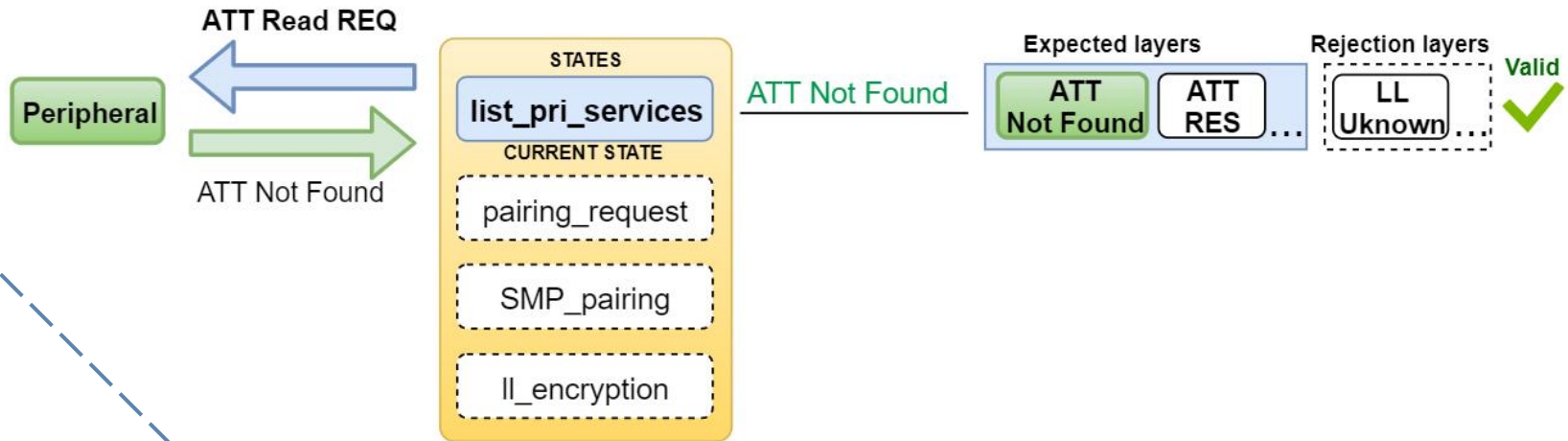
NXP LLID Deadlock (CVE-2019-17060)

# Validation Strategy - Exemplified



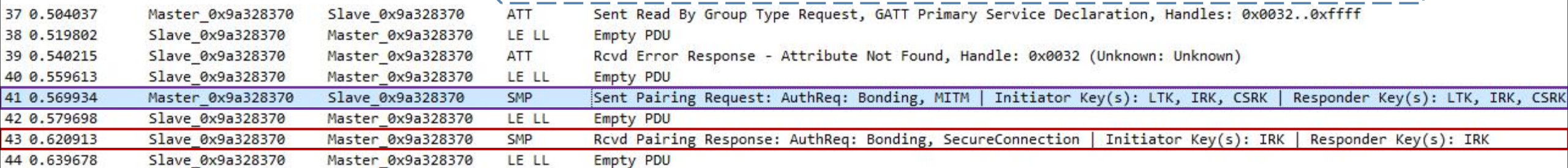
**Packet Validation**

Mixed Valid and invalid responses

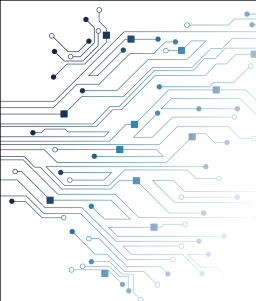


37	0.504037	Master_0x9a328370	Slave_0x9a328370	ATT	Sent Read By Group Type Request, GATT Primary Service Declaration, Handles: 0x0032..0xffff
38	0.519802	Slave_0x9a328370	Master_0x9a328370	LE LL	Empty PDU
39	0.540215	Slave_0x9a328370	Master_0x9a328370	ATT	Rcvd Error Response - Attribute Not Found, Handle: 0x0032 (Unknown: Unknown)
40	0.559613	Slave_0x9a328370	Master_0x9a328370	LE LL	Empty PDU

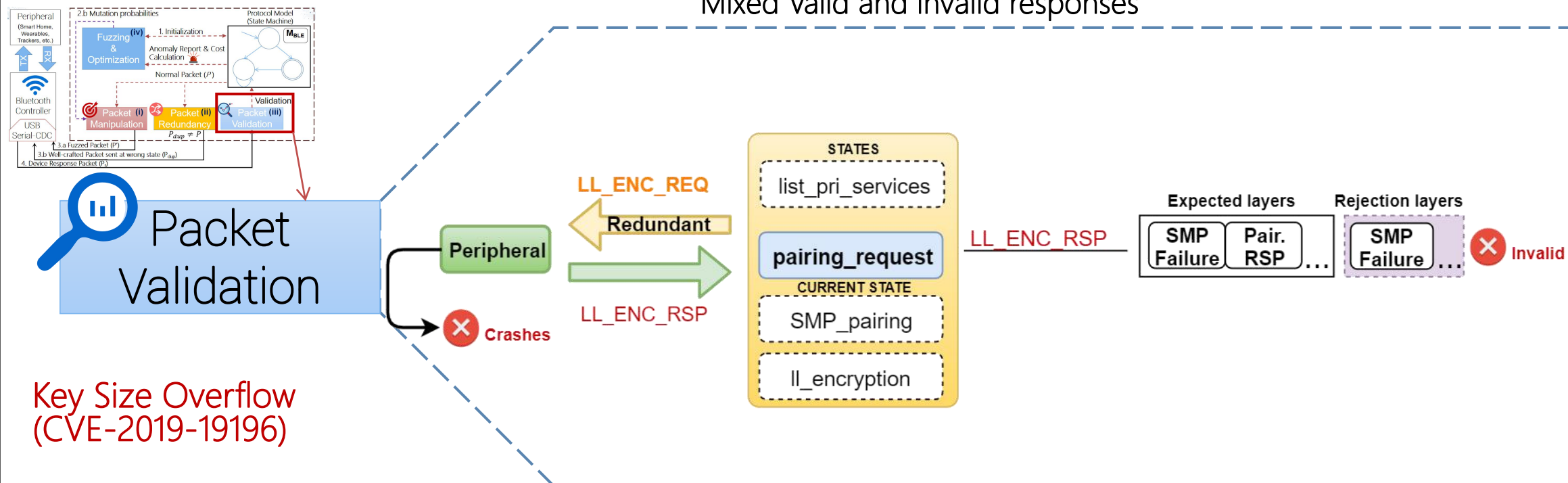
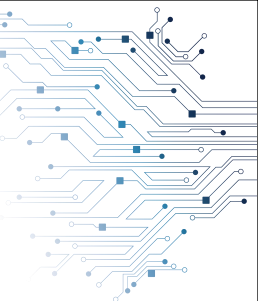
# Validation Strategy - Exemplified







# Validation Strategy - Exemplified

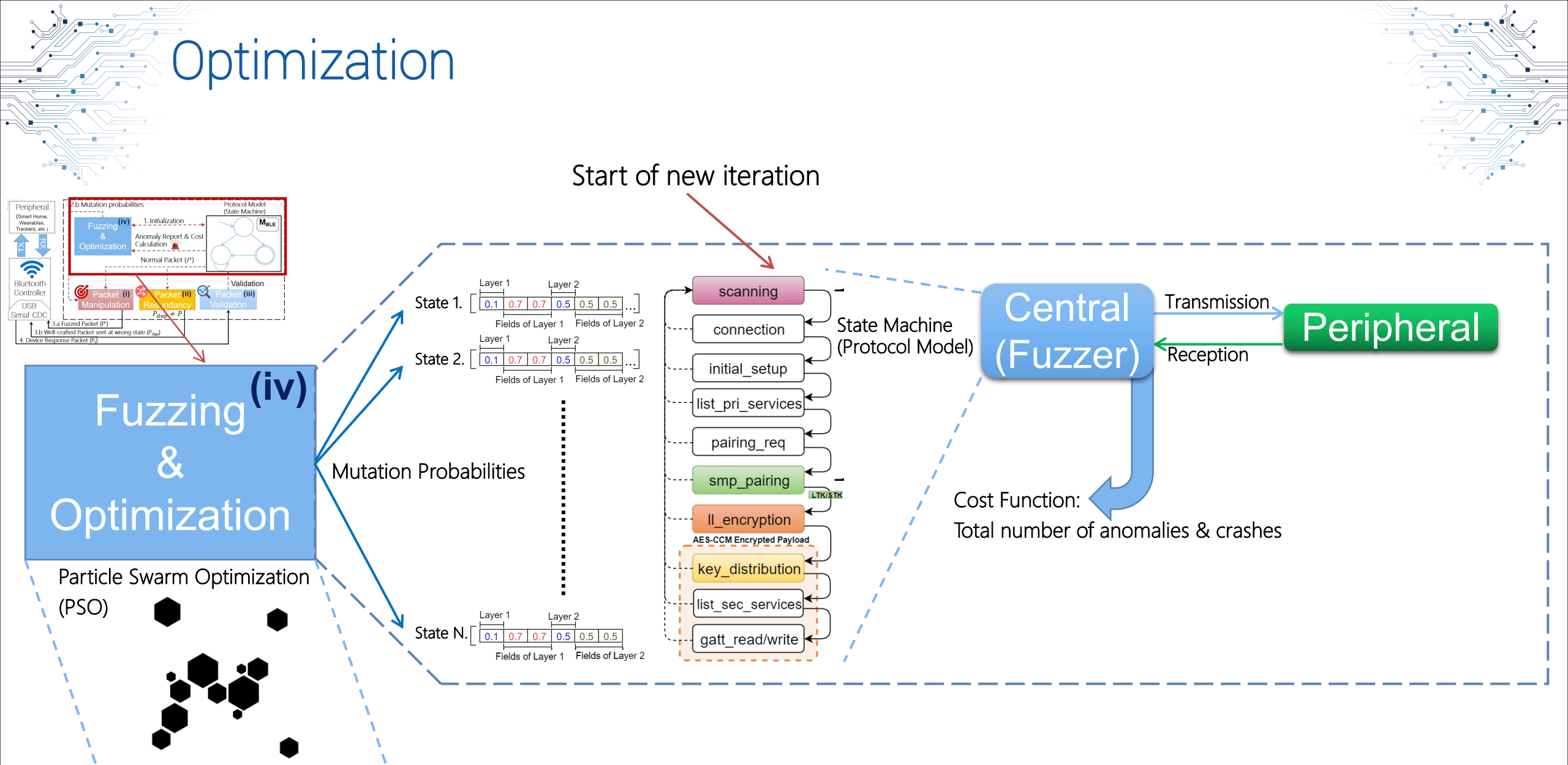


## Key Size Overflow (CVE-2019-19196)

37	0.504037	Master_0x9a328370	Slave_0x9a328370	ATT	Sent Read By Group Type Request, GATT Primary Service Declaration, Handles: 0x0032..0xffff
38	0.519802	Slave_0x9a328370	Master_0x9a328370	LE LL	Empty PDU
39	0.540215	Slave_0x9a328370	Master_0x9a328370	ATT	Rcvd Error Response - Attribute Not Found, Handle: 0x0032 (Unknown: Unknown)
40	0.559613	Slave_0x9a328370	Master_0x9a328370	LE LL	Empty PDU
41	0.569934	Master_0x9a328370	Slave_0x9a328370	SMP	Sent Pairing Request: AuthReq: Bonding, MITM   Initiator Key(s): LTK, IRK, CSRK   Responder Key(s): LTK, IRK, CSRK
42	0.579698	Slave_0x9a328370	Master_0x9a328370	LE LL	Empty PDU
43	0.620913	Slave_0x9a328370	Master_0x9a328370	SMP	Rcvd Pairing Response: AuthReq: Bonding, SecureConnection   Initiator Key(s): IRK   Responder Key(s): IRK
44	0.639678	Slave_0x9a328370	Master_0x9a328370	LE LL	Empty PDU
45	0.648681	Master_0x9a328370	Slave_0x9a328370	LE LL	Control Opcode: LL_ENC_REQ
46	0.660411	Slave_0x9a328370	Master_0x9a328370	LE LL	Empty PDU
47	0.680397	Slave_0x9a328370	Master_0x9a328370	LE LL	Control Opcode: LL_ENC_RSP
48	0.699994	Slave_0x9a328370	Master_0x9a328370	LE LL	Empty PDU

22

# Optimization

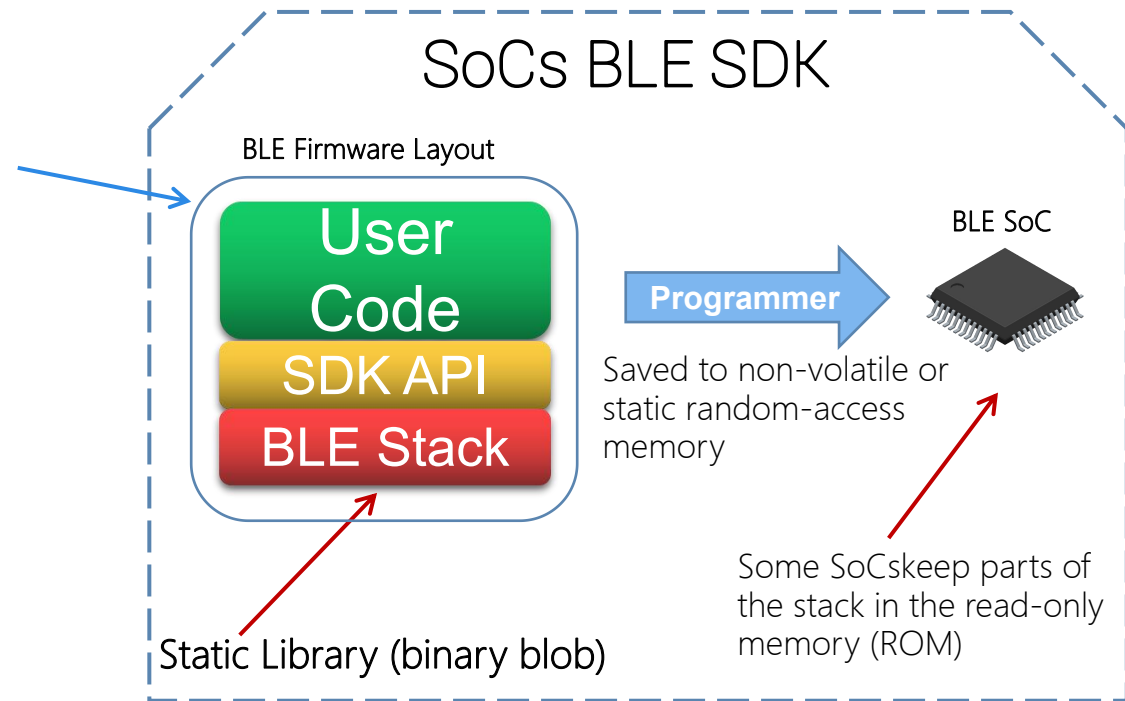




# Evaluation - Setup

Table 1: Development Platforms used for evaluation

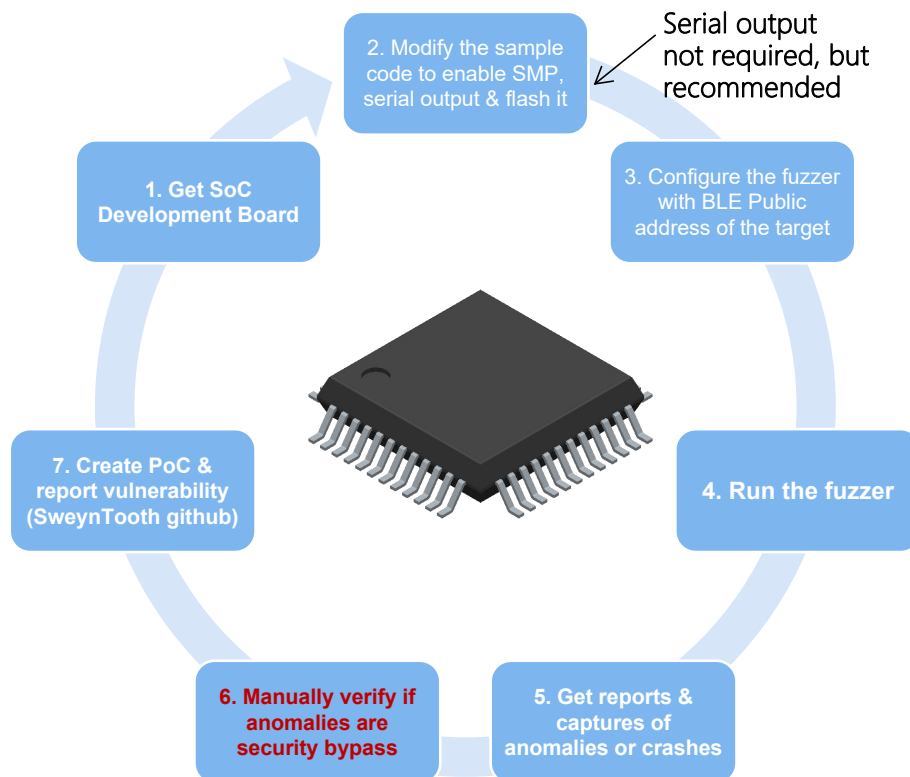
Silicon Vendor	Development Platform	BLE Ver.	Sample Code Name
Cypress (PSoC 6)	CY8CPROTO-63	5.0	Device_Information_Service
Cypress (PSoC 4)	CY5677	4.2	Device_Information_Service
Texas Instruments	LaunchXL-CC2640R2	5.0	project_zero
Texas Instruments	CC2540EMK-USB	4.1	simple_peripheral
Telink	TLSR8258 USB	5.0	8258_ble_sample
STMicroelectronics	NUCLEO-WB55	5.0	BLE_BloodPressure
STMicroelectroncis	STEVAL-IDB008V2	5.0	SlaveSec_A0
NXP	USB-KW41Z	4.2	heart_heart_rate_sensor_bm
Dialog	DA14681DEVKIT	4.2	ble_adv
Dialog	DA14580DEVKIT	4.1	ble_app_peripheral
Microchip	SAMB11 Xplained	4.1	blood_pressure_samb11
Nordic Semi.	nRF51 Dongle	5.0	ble_app_hrs
Nordic Semi.	nRF52840 Dongle	5.0	ble_app_gatts_c



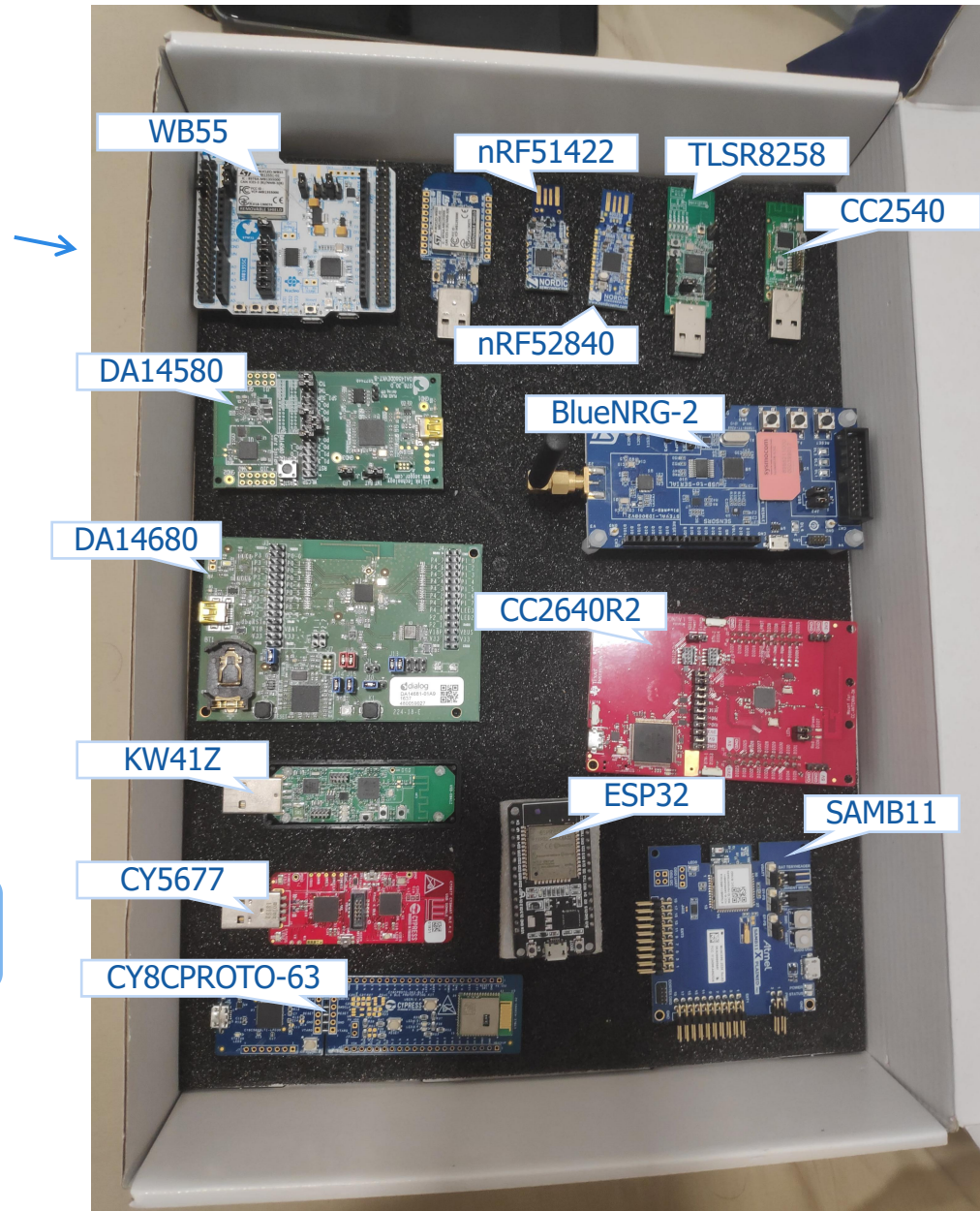
# Evaluation - Setup

Table 1: Development Platforms used for evaluation

Silicon Vendor	Development Platform	BLE Ver.	Sample Code Name
Cypress (PSoC 6)	CY8CPROTO-63	5.0	Device_Information_Service
Cypress (PSoC 4)	CY5677	4.2	Device_Information_Service
Texas Instruments	LaunchXL-CC2640R2	5.0	project_zero
Texas Instruments	CC2540EMK-USB	4.1	simple_peripheral
Telink	TLSR8258 USB	5.0	8258_ble_sample
STMicroelectronics	NUCLEO-WB55	5.0	BLE_BloodPressure
STMicroelectroncis	STEVAL-IDB008V2	5.0	SlaveSec_A0
NXP	USB-KW41Z	4.2	heart_heart_rate_sensor_bm
Dialog	DA14681DEVKIT	4.2	ble_adv
Dialog	DA14580DEVKIT	4.1	ble_app_peripheral
Microchip	SAMB11 Xplained	4.1	blood_pressure_samb11
Nordic Semi.	nRF51 Dongle	5.0	ble_app_hrs
Nordic Semi.	nRF52840 Dongle	5.0	ble_app_gatts_c



## Target BLE SoC Dev. Kits



# Evaluation - Comparison

Qualitative comparison with publicly available tools

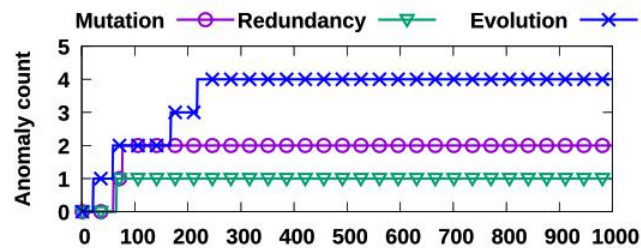
Comparison		
Tools	Supported Layer(s)	Fuzzing Strategy
Stack Smasher	L2CAP	Random
BLEFuzz	ATT	Random / Handcrafted
bfuzz (IotCube)	L2CAP	Random / Test database
Our Fuzzer	LL / L2CAP / SMP / ATT	Evolutionary

- BT Classic only. Adaption was needed for comparison;
- Only a subset of L2CAP is available for BLE;
- Previous Bluetooth fuzzers detect crashes, but not logic problems (anomalies);
- Link Layer was not supported by other fuzzers.

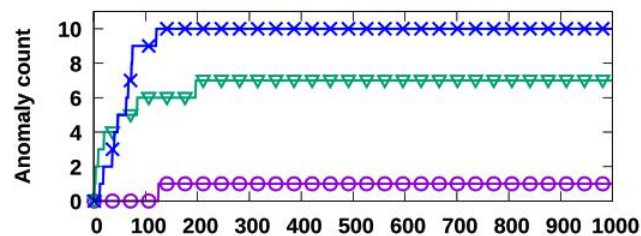


# Evaluation

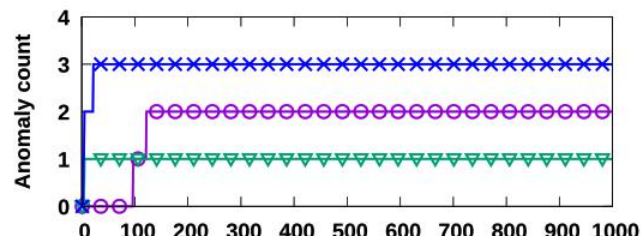
## Anomalies vs iteration



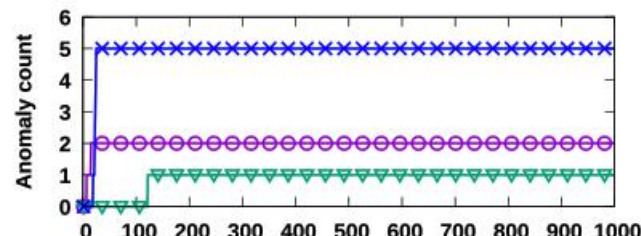
a) Fuzzing Iterations in TI CC2640



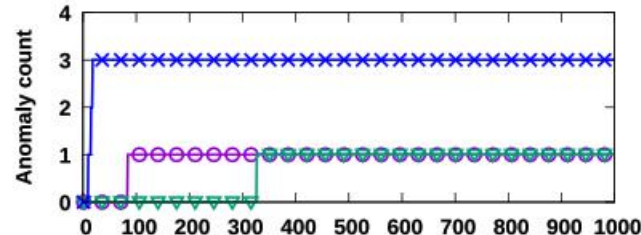
b) Fuzzing Iterations in Telink



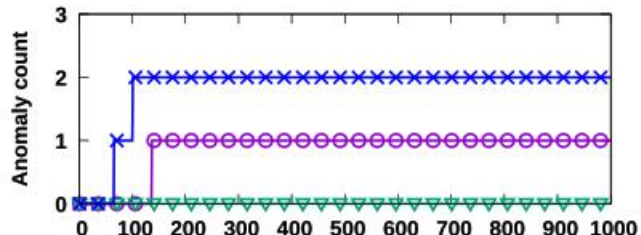
c) Fuzzing Iterations in WB55



e) Fuzzing Iterations in NXP KW41Z



f) Fuzzing Iterations in PSoC 6



g) Fuzzing Iterations in DA14680

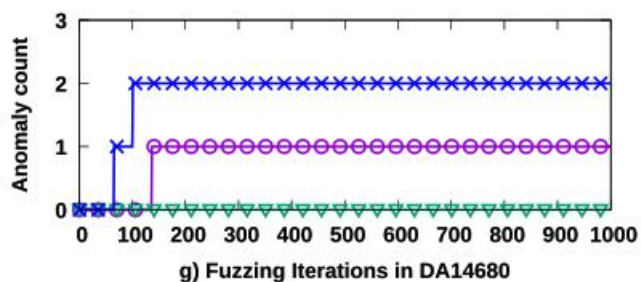
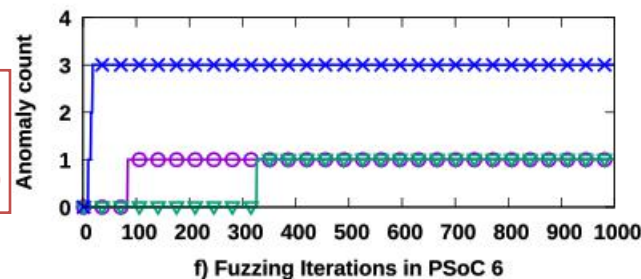
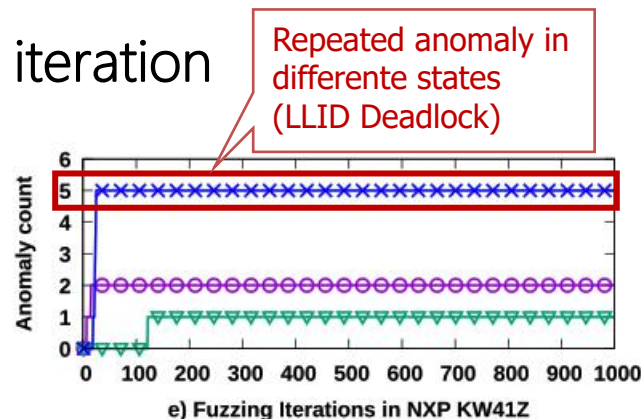
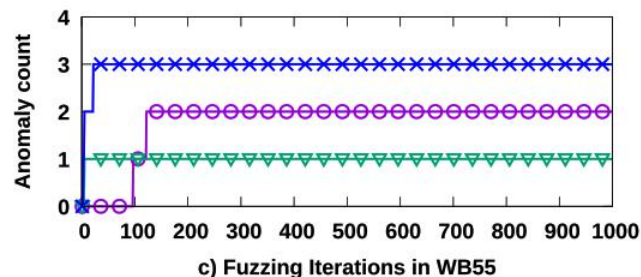
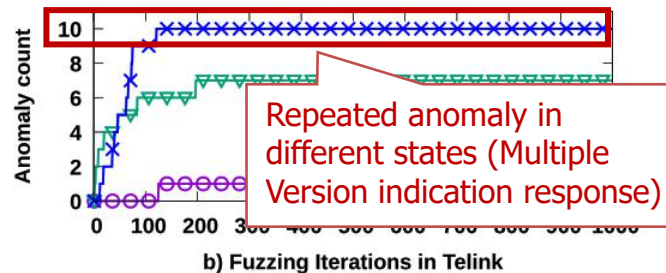
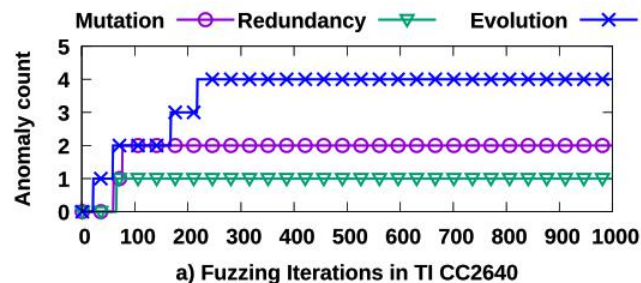
## Summary of Evaluation Time for Each Device (\*channel hop Interval = 20ms)

Platform	Iterations	Total Time	1st Crash	1st Anomaly	Model Coverage
CY8CPROTO-63	1000	1 h. 06 min.	1 min.	<1 min.	27 (50.0%)
CY5677	1000	2 h. 27 min.	<1 min.	8 min.	29 (53.7%)
USB-KW41Z	1000	1 h. 30 min.	<1 min.	2 min.	24 (44.4%)
DA14681DEVKIT	1000	1 h. 16 min.	10 min.	6 min.	30 (55.5%)
DA14580DEVKIT	1000	2 h. 7 min.	5 min.	1 min.	32 (59.3%)
CC2640R2 Devkit	1000	1 h. 57 min.	4 min.	1 min.	31 (57.40%)
CC2540 Devkit	1000	1 h. 37 min.	2 min.	19 min.	34 (62.96%)
Nucleo-WB55	1000	1 h. 45 min.	<1 min.	2 min.	26 (48.15%)
BlueNRG-2	1000	1 h. 14 min.	<1 min.	9 min.	30 (55.55%)
ATSAMB11	1000	2 h. 39 min.	2 min.	10 min.	33 (61.1%)
TLSR8258	1000	1 h. 56 min.	5 min.	<1 min.	36 (66.67%)

\*Same as Connection Interval

# Evaluation

## Anomalies vs iteration



Repeated anomaly in different states (LLID Deadlock)

Repeated anomaly in different states (Multiple Version indication response)

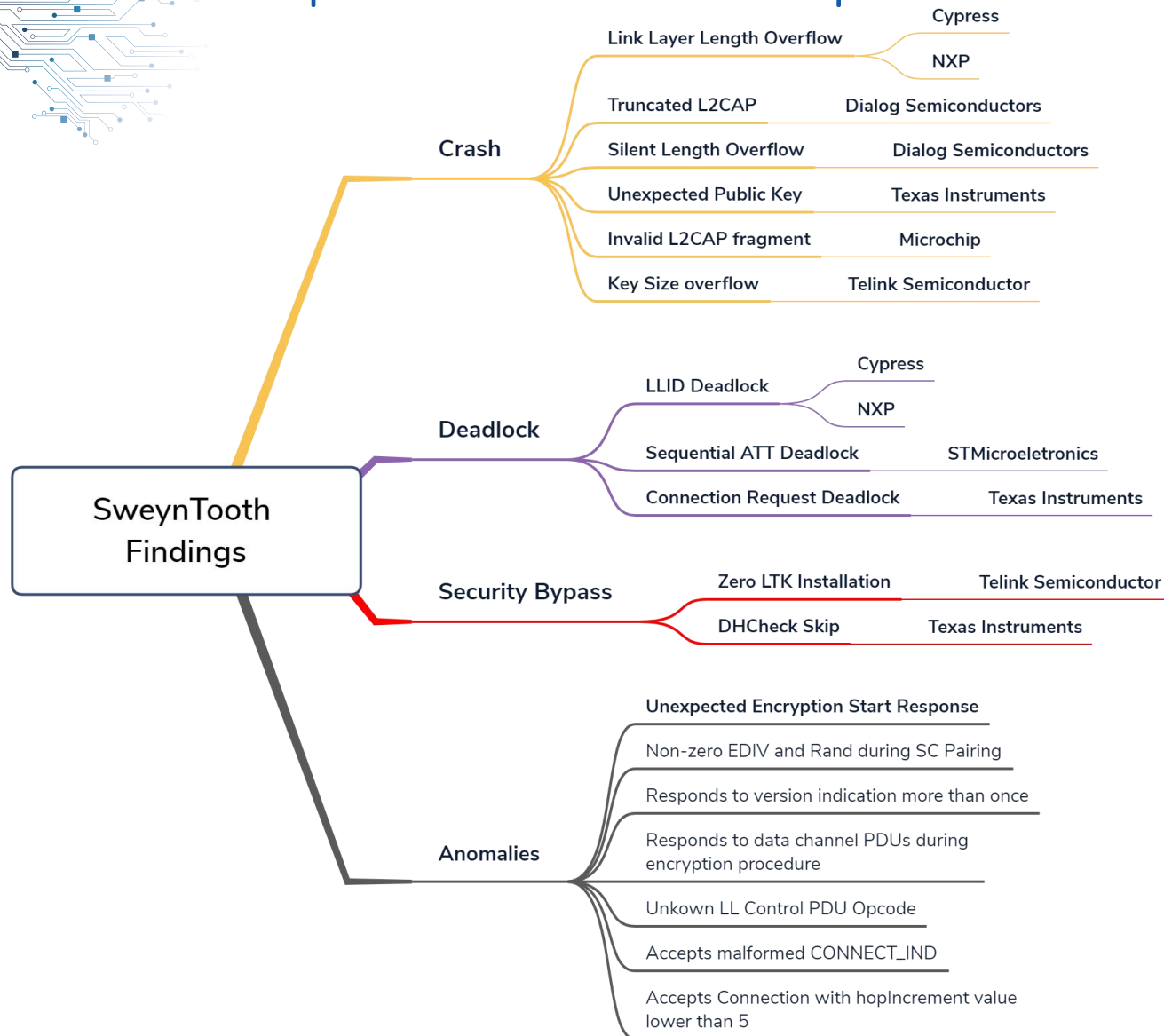
## Summary of Evaluation Time for Each Device (\*channel hop Interval = 20ms)

Platform	Iterations	Total Time	1st Crash	1st Anomaly	Model Coverage
CY8CPROTO-63	1000	1 h. 06 min.	1 min.	<1 min.	27 (50.0%)
CY5677	1000	2 h. 27 min.	<1 min.	8 min.	29 (53.7%)
USB-KW41Z	1000	1 h. 30 min.	<1 min.	2 min.	24 (44.4%)
DA14681DEVKIT	1000	1 h. 16 min.	10 min.	6 min.	30 (55.5%)
DA14580DEVKIT	1000	2 h. 7 min.	5 min.	1 min.	32 (59.3%)
CC2640R2 Devkit	1000	1 h. 57 min.	4 min.	1 min.	31 (57.40%)
CC2540 Devkit	1000	1 h. 37 min.	2 min.	19 min.	34 (62.96%)
Nucleo-WB55	1000	1 h. 45 min.	<1 min.	2 min.	26 (48.15%)
BlueNRG-2	1000	1 h. 14 min.	<1 min.	9 min.	30 (55.55%)
ATSAMB11	1000	2 h. 39 min.	2 min.	10 min.	33 (61.1%)
TLSR8258	1000	1 h. 56 min.	5 min.	<1 min.	36 (66.67%)

\*Same as Connection Interval



# Impact - Non-compliance in the wild!

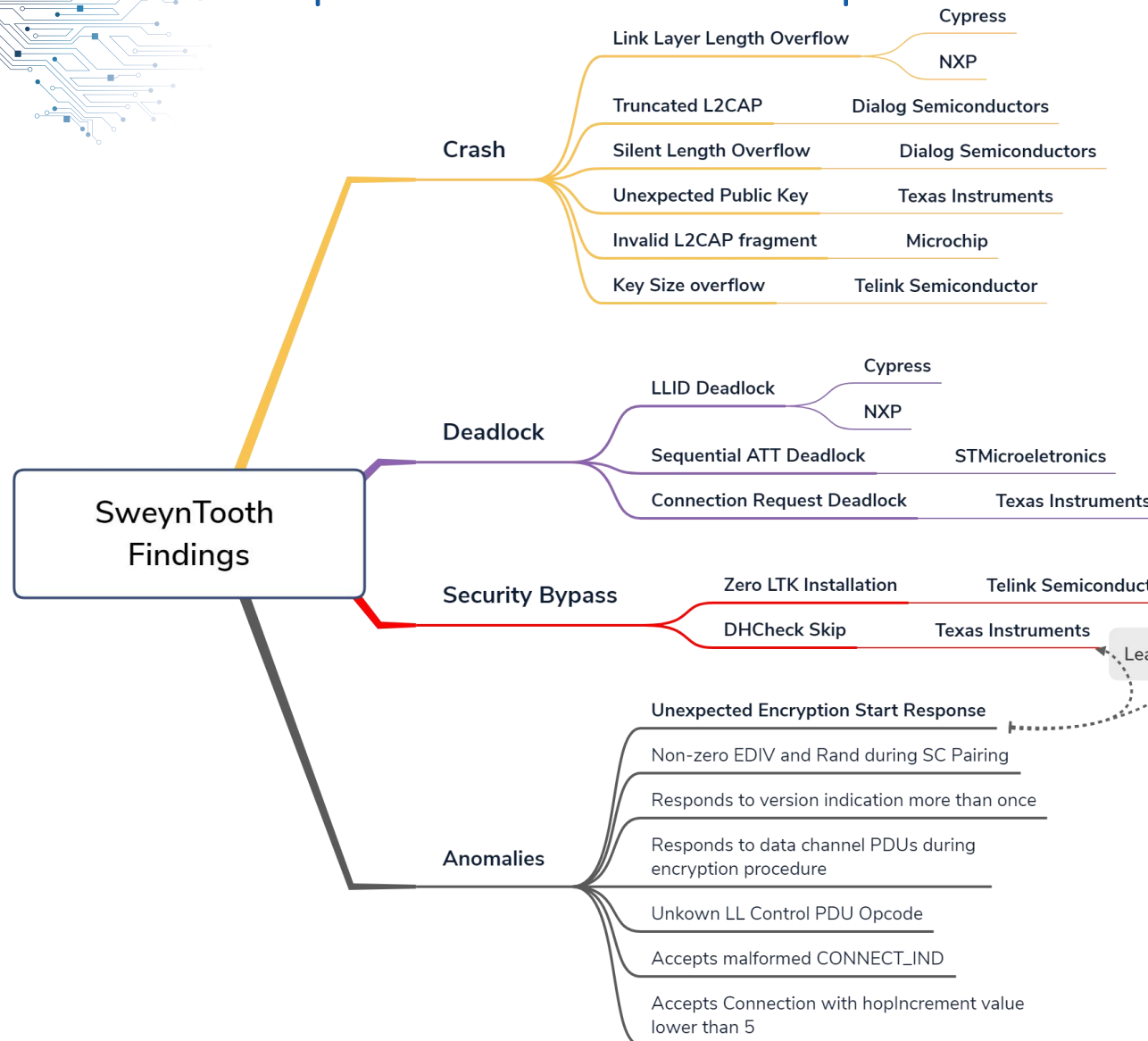


1st SweynTooth disclosure  
(9th February, 2020)  
\*DHCheck reported later

\*Details of all vulnerabilities & non-compliances on <https://asset-group.github.io/disclosures/sweyntooth/>

\*Test scripts are available on [https://github.com/Matheus-Garbelini/sweyntooth\\_bluetooth\\_low\\_energy\\_attacks/tree/master/extras](https://github.com/Matheus-Garbelini/sweyntooth_bluetooth_low_energy_attacks/tree/master/extras)

# Impact - Non-compliance in the wild!



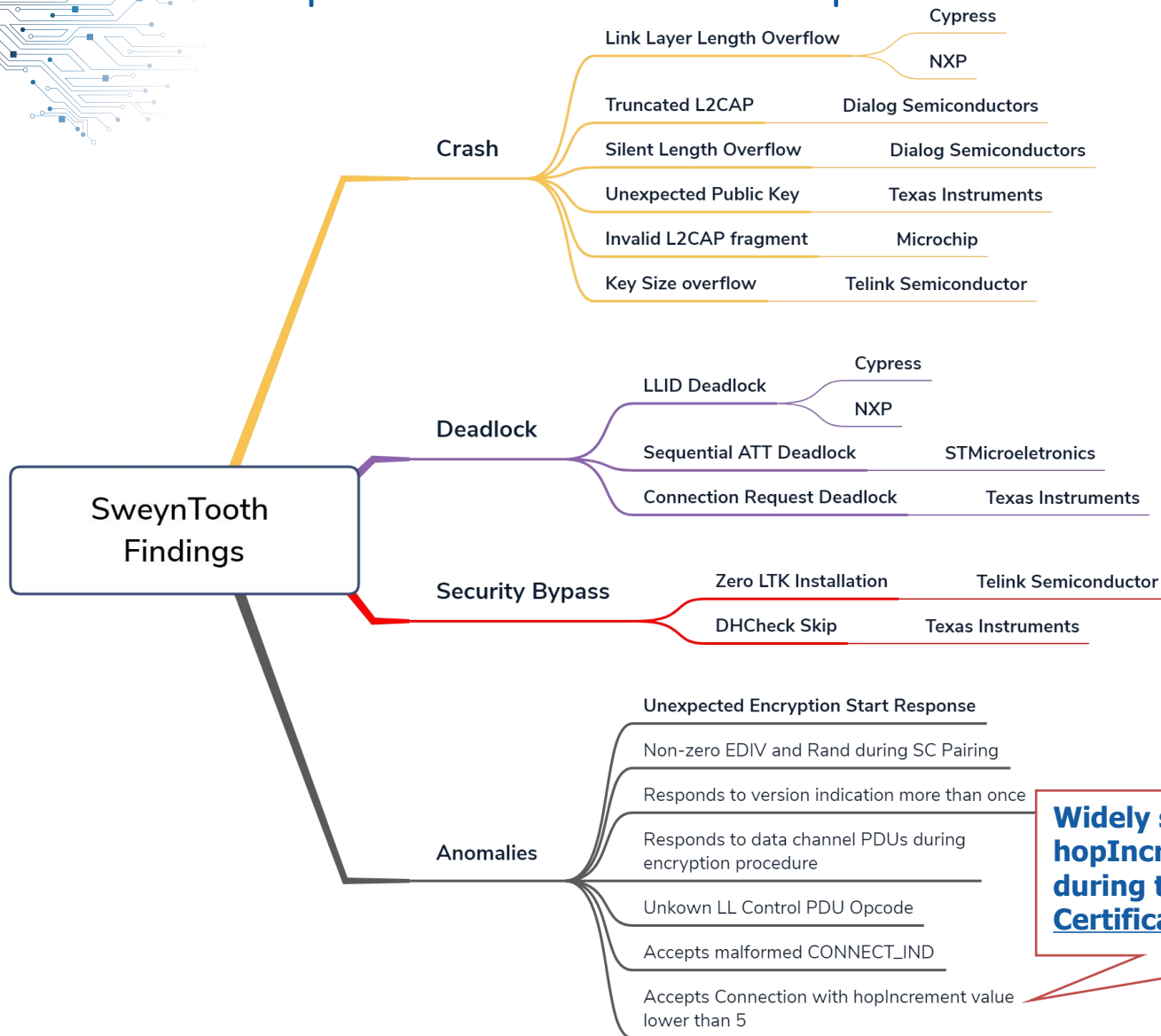
1st SweynTooth disclosure  
(9th February, 2020)  
\*DHCheck reported later

**Security bypass issues were found to be a mishandling of A1 - Encryption setup happens during SMP pairing procedure**

\*Details of all vulnerabilities & non-compliances on <https://asset-group.github.io/disclosures/sweyntooth/>

\*Test scripts are available on [https://github.com/Matheus-Garbelini/sweyntooth\\_bluetooth\\_low\\_energy\\_attacks/tree/master/extras](https://github.com/Matheus-Garbelini/sweyntooth_bluetooth_low_energy_attacks/tree/master/extras)

# Impact - Non-compliance in the wild!



1st SweynTooth disclosure  
(9th February, 2020)  
\*DHCheck reported later

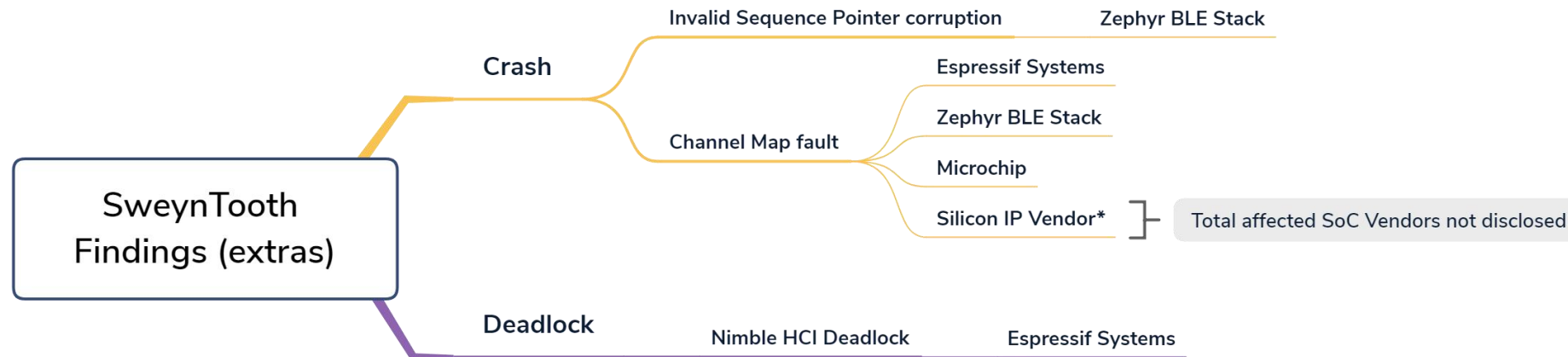
**Widely spread non-compliance.  
hopIncrement is a fundamental field used  
during the connection procedure.  
Certification didn't catch it?**

\*Details of all vulnerabilities & non-compliances on <https://asset-group.github.io/disclosures/sweyntooth/>

\*Test scripts are available on [https://github.com/Matheus-Garbelini/sweyntooth\\_bluetooth\\_low\\_energy\\_attacks/tree/master/extras](https://github.com/Matheus-Garbelini/sweyntooth_bluetooth_low_energy_attacks/tree/master/extras)

# Impact - Non-compliance in the wild!

2nd SweynTooth disclosure  
(13th July, 2020)

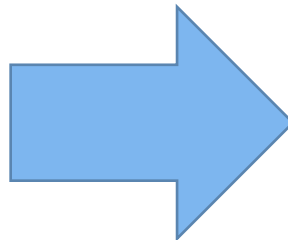
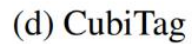
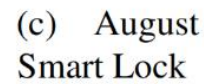
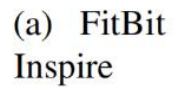


\*Details of all vulnerabilities & non-compliances on <https://asset-group.github.io/disclosures/sweyntooth/>

\*Test scripts are available on [https://github.com/Matheus-Garbelini/sweyntooth\\_bluetooth\\_low\\_energy\\_attacks/tree/master/extras](https://github.com/Matheus-Garbelini/sweyntooth_bluetooth_low_energy_attacks/tree/master/extras)



# Some affected IoT products



# Impact

## CubiTag: Public Key Crash (Deadlock)

The image is a composite of four parts. On the left, a Windows File Explorer window shows a folder named 'sequential\_crt\_deadlock.py' with a file 'trigger\_exploit\_August\_Smartibat' dated 11/02/2020 10:58. Below it, a Windows Command Prompt window shows a Python script running, which attempts to connect to a Bluetooth device and eventually crashes with a 'KeyboardInterrupt' error. On the right, a mobile app interface shows a map with a location pin and a 'CubiTag' label. Below the app interface, a person is shown using a screwdriver to pry open the back of a smartphone.

The user must remove and  
reinsert the batteries with a  
screwdriver to reboot the device  
and make it work again

[https://youtu.be/lw8sIBLWE\\_w](https://youtu.be/lw8sIBLWE_w)

**The user must remove and reinsert the batteries with a screwdriver to reboot the device and make it work again**

<https://youtu.be/lw8sIBLWE> w

# Disclosure process

Asset Research Website  
Public Disclosure  
(9th February, 2020)



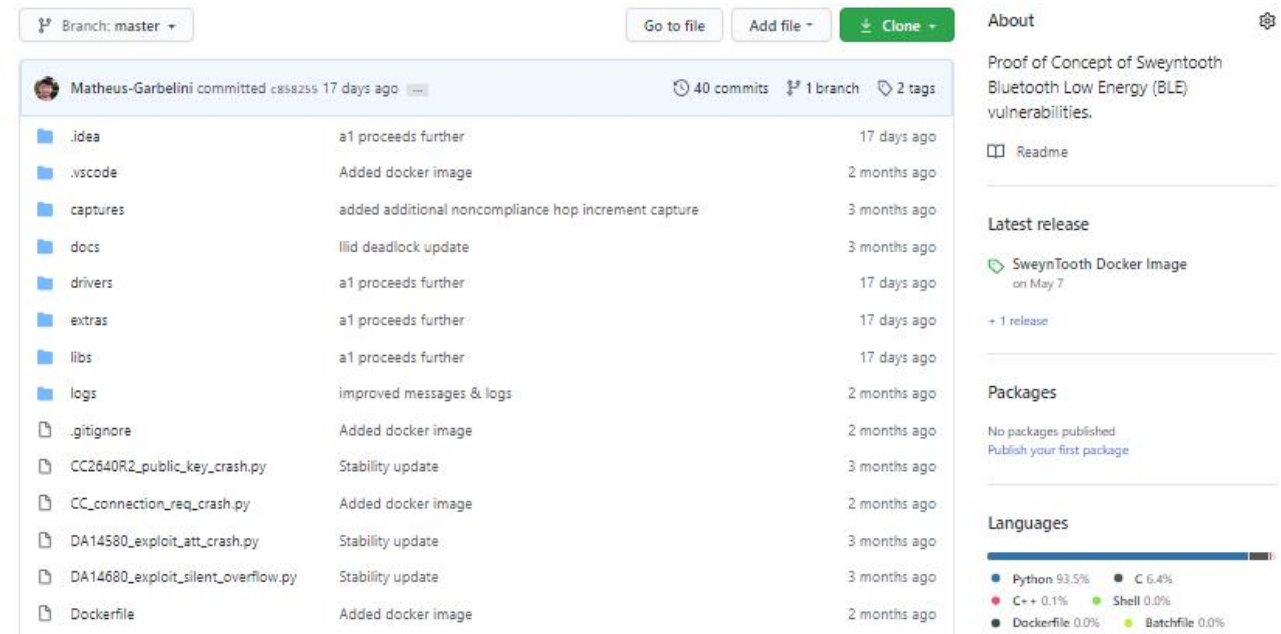
## Unleashing Mayhem over Bluetooth Low Energy

Matheus E. Garbelini<sup>1</sup>; Sudipta Chattopadhyay<sup>1</sup>; Chundong Wang<sup>1</sup>

<sup>1</sup>Singapore University of Technology and Design

## Exploits repository (GitHub)

[https://github.com/Matheus-Garbelini/sweyntooth\\_bluetooth\\_low\\_energy\\_attacks](https://github.com/Matheus-Garbelini/sweyntooth_bluetooth_low_energy_attacks)



- Disclosure window of 90 days, starting since the last communicated SoC vendor;
- Second batch of SweynTooth vulnerabilities privately shared in advance with CSA and HSA, Singapore;
- Bluetooth SIG has also requested early access to the non-disclosed information of the 2nd batch (13th July, 2020);
- As far as we now, only one vendor has yet to create a firmware patch!





# Conclusion

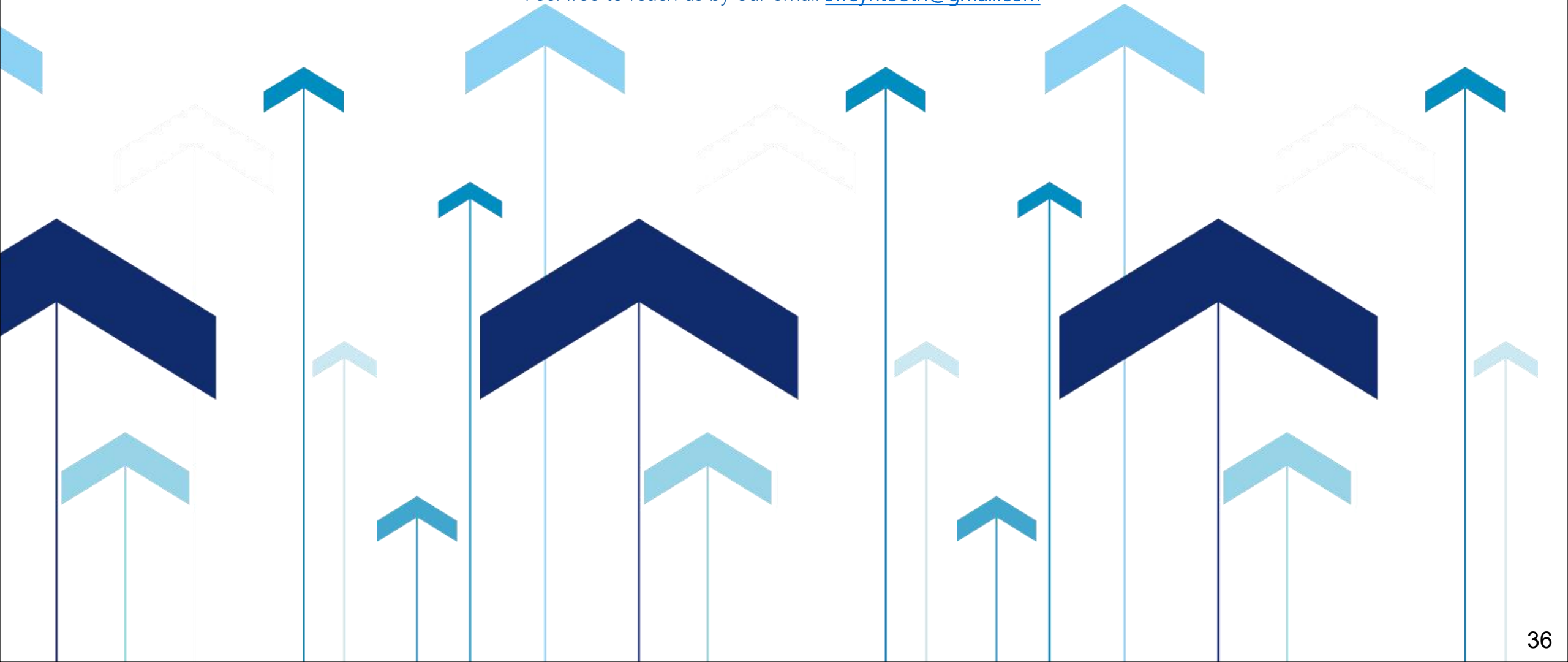


- Certification **does not** prevent against bad implementation **nor guarantee** an BLE stack to be free of non-compliances.
- Procedures which conflict with each other could be better clarified on the standard (i.g., unexpected encryption response) to avoid related security bypass attacks.
- Over-the-air fuzzing is still a good way to find many wireless bugs, given it a proper control over the lowest layers of the target wireless protocol.
- What about other wireless technologies? BLE Mesh, Wi-Fi EasyMesh, 5G, NB-IoT? More fuzzing tool are needed.
- Lesson learned. Product vendors may rethink their solution and give it more priority for SoC vendors with greater security response and easier patching process.
- The fuzzer is available open source upon request to [sweyntooth@gmail.com](mailto:sweyntooth@gmail.com)



# Thank you Questions?

Feel free to reach us by our email [sweyntooth@gmail.com](mailto:sweyntooth@gmail.com)



# Final Remark: Get Ready for BLE Experimentation

## What if I want to experiment with BLE myself?

Simplest Setup: Scapy Python API to get you started with BLE experimentation is available on our GitHub repo. Use of our custom firmware requires a nRF52840 Dongle (~10-11USD). Works on Linux, OSX and Windows distros.



### Crafting and Sending a Link Layer Packet example:

```
# Send LL version indication request
pkt = BTLE(access_addr=access_address) / BTLE_DATA() / CtrlPDU() / LL_VERSION_IND(version='4.2')
driver.send(pkt)
```

### Receiving a Link Layer Packet example:

```
while True:
    # Receive and decode packet from the NRF52 Dongle
    pkt = BTLE(driver.raw_receive())
    # Check peripheral version
    if pkt and LL_VERSION_IND in pkt:
        print('Peripheral version:' + str(pkt.version))
```