


BLUFFS: Bluetooth Forward and Future Secrecy Attacks and Defenses



 37C3 (2023)

[Daniele Antonioli](#)
([EURECOM](#), [S3](#))



ORSHIN

Ciao! I am Daniele Antonioli

- Prof at EURECOM (French riviera, 🏖️, 🏔️)
- Research in *applied system security and privacy*
 - Wireless (Bluetooth, Wi-Fi, proprietary, ...)
 - Embedded (IoT, cars, ...)
 - Mobile (Android, iOS, ...)
- I am hiring PhDs and Postdocs
 - Talk to me later
 - Email me: antonioli.daniele@gmail.com
 - Visit my website: <https://francozappa.github.io>



Talk based on my ACM CCS'23 [paper](#)

BLUFFS: Bluetooth Forward and Future Secrecy Attacks and Defenses

Daniele Antonioli

EURECOM

Sophia Antipolis, France

daniele.antonioli@eurecom.fr

ABSTRACT

Bluetooth is a pervasive technology for wireless communication. Billions of devices use it in sensitive applications and to exchange private data. The security of Bluetooth depends on the Bluetooth standard and its two security mechanisms: pairing and session establishment. No prior work, including the standard itself, analyzed the *future and forward secrecy* guarantees of these mechanisms, e.g., if Bluetooth pairing and session establishment defend past

KEYWORDS

Bluetooth, forward secrecy, future secrecy, attacks, defenses

ACM Reference Format:

Daniele Antonioli. 2023. BLUFFS: Bluetooth Forward and Future Secrecy Attacks and Defenses. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security (CCS '23)*, November 26–30, 2023, Copenhagen, Denmark. ACM, New York, NY, USA, 15 pages. <https://doi.org/10.1145/3576915.3623066>

Talk contributes to the [ORSHIN](#) EU Research Grant

ORSHIN: Open-source ReSilient Hardware and software for Internet of thiNgs

How to design embedded and connected devices taking advantage of open source hardware (and software)

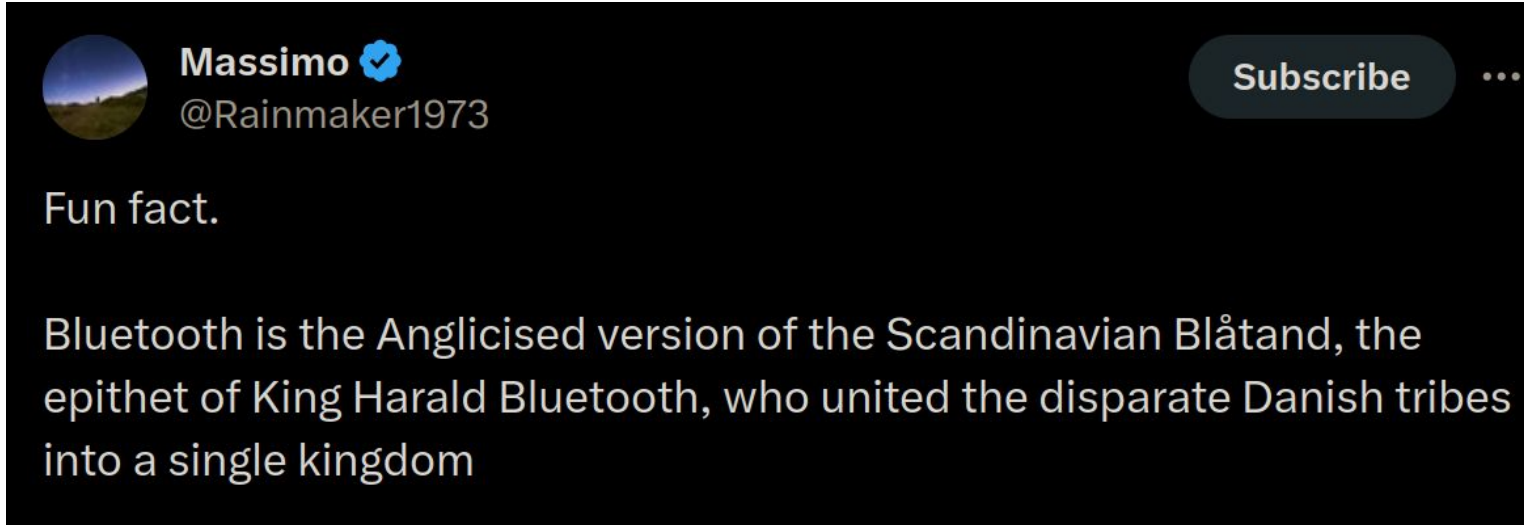


Introduction

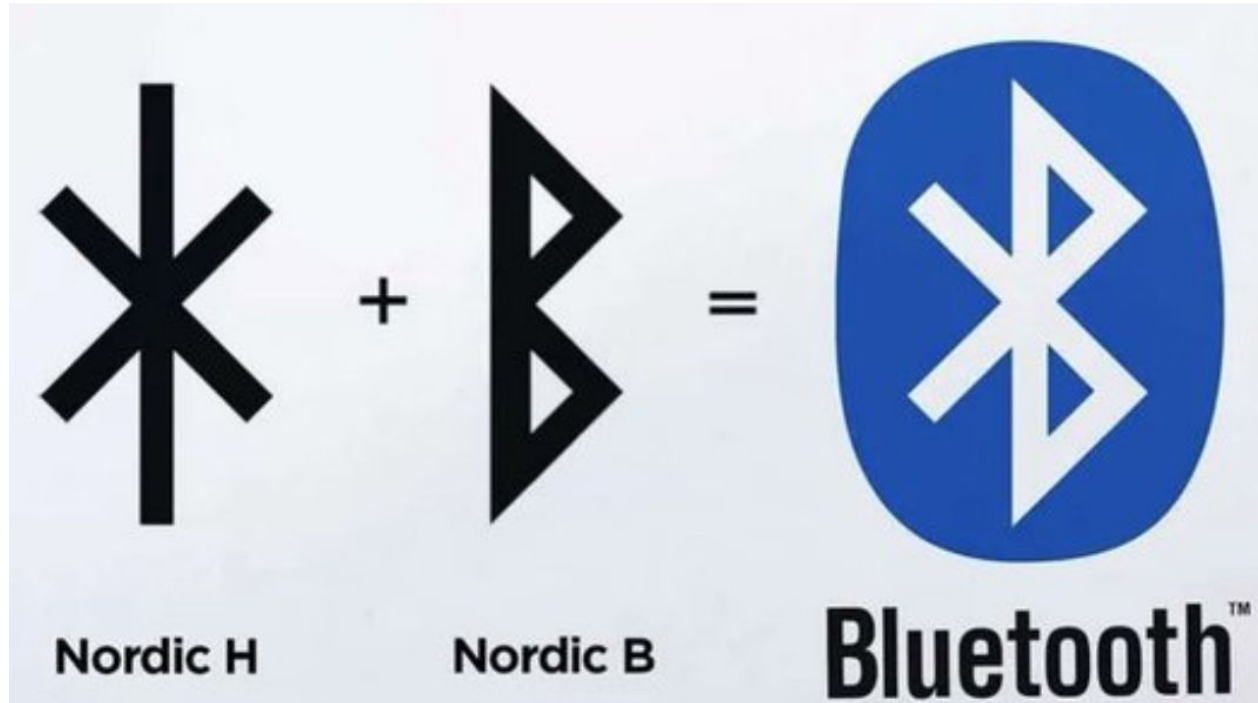
Bluetooth (BT)

- BT is a pervasive low-power wireless technology
 - Specified in [bluetooth-core.pdf \(v5.4\)](#)
 - **BC: Bluetooth Classic** (high throughput)
 - **BLE: Bluetooth Low Energy** (very low power)
 - Interoperable aka used by [billions of heterogeneous devices](#), e.g., smartphones, laptops, cars, wearables, sensors, medical, ...

BT Name ([ref](#))



BT Logo ([ref](#))



BT Specification ([ref](#))

- BT specification
 - Defines technologies to create *interoperable* BT devices
 - Transports: [BC](#), [BLE](#), ...
 - Components: Host, Controller, HCI, ...
 - Security: Pairing, Session establishment, ...
- **One BT spec vulnerability → Billions of exploitable devices**
 - 2021: **BLUR cross-transport overwrites** on [BC](#) and [BLE](#)
 - 2020: **BIAS authentication bypasses** on [BC](#)
 - 2019: **KNOB key downgrades** on [BC](#) and [BLE](#)

BT Security

- Pairing
 - *Pairing key (PK)*, long term, BLE entropy negotiation
 - Optionally authenticated (numeric PIN, ...)
- Session Establishment
 - *Session key (SK)*, fresh, BC entropy negotiation
 - $SK = \text{kdf}(PK, \text{pars})$
- Negotiable security mode
 - Secure Connections (SC)
 - Legacy Secure Connections (LSC)

Forward and Future Secrecy (FoS, FuS)

- Forward Secrecy (FoS)
 - Protects **past** sessions against **key** compromise
 - Eg: **key** = HKDF(const, key_past)
- Future Secrecy (FuS)
 - Protects **future** sessions against **key** compromise
 - Eg: **key_future** = HKDF(dhss, key)

BT FoS and FuS?

- **Not** discussed in the BT specification
- **No prior** evaluations (academia, industry, ...)
- Despite **widespread** real-world usage (TLS1.3, Signal, ...)
- **BLUFFS research** fills this relevant gap!



BLUFFS Contributions

- First study on BT FoS and FuS ([paper](#), [slides](#))
- Uncover 2 FoS/FuS vulns in BC SK derivation
- Develop 6 BLUFFS attacks breaking BC sessions' FoS/FuS
- Exploit 18 popular devices (Intel, Broadcom, Apple, Google, Microsoft, CSR, Logitech, Infineon, Bose, Dell, Xiaomi, ...)
- Fix the attacks with a compliant and practical protocol
- Report critical findings to BT SIG, got [CVE-2023-24023](#)
- Release [BLUFFS toolkit](#) to test the attacks and BC FoS/FuS

BLUFFS Attacks



BLUFFS Threat model

- **BC should** provide **FoS** and **FuS** among sessions
 - long term PK is not compromised
 - fresh SK derivation is not vulnerable
- Alice (Central) and Bob (Peripheral)
 - Share PK
 - Use SC or LSC
- **Charlie (attacker)**
 - Model: proximity-based, cannot compromise PK or all SKs
 - Goals: break sessions' **FoS** and **FuS**
 - Impact: impersonate and MitM devices across sessions



BLUFFS Attacks

t_0 : Alice and Bob establish PK

t_1 : Charlie forces **weak SK_C** , saves **SK_C** kdf pars, sniffs s_{t_1}, \dots

t_2 : Charlie brute forces **SK_C** and **breaks s_{t_1}, \dots, s_{t_2}** (breaks **FoS**)

t_3 : Charlie re-forces **SK_C** and **breaks s_{t_3}, s_{t_4}, \dots** (breaks **FuS**)

BLUFFS Attacks



t_0 : Alice and Bob establish PK

t_1 : Charlie forces **weak SK_C** , saves **SK_C** kdf pars, sniffs s_{t_1}, \dots

t_2 : Charlie brute forces **SK_C** and **breaks s_{t_1}, \dots, s_{t_2}** (breaks **FoS**)

t_3 : Charlie re-forces **SK_C** and **breaks s_{t_3}, s_{t_4}, \dots** (breaks **FuS**)

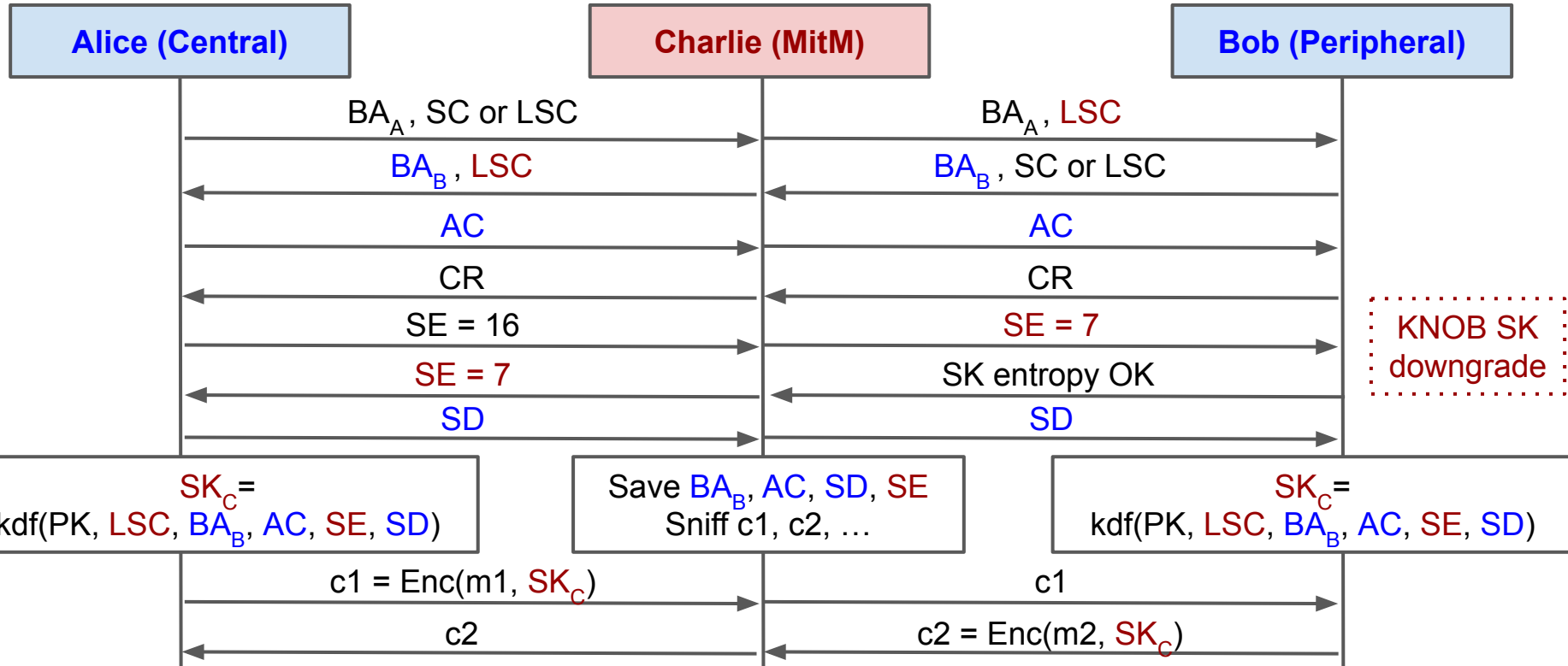
t_∞ : Charlie **celebrates (One More Time)!**

Daft Punk Detour

- French electronic music duo
 - Formed in Paris 1993
- Music combining multiple genres
 - Like BLUFFS combining multiple tricks
- Negatively seen by conservative folks, but *new and relevant*
 - Like BLUFFS and other off-the-wall efforts



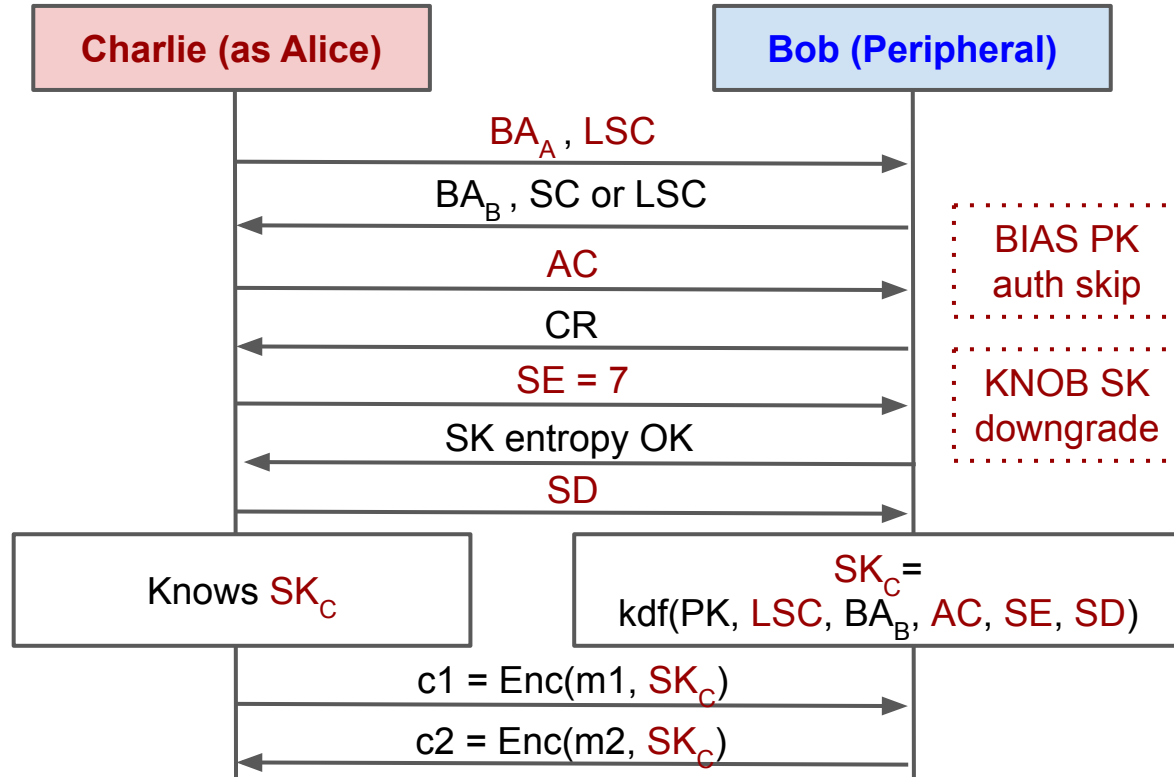
t_1 : Force **weak** SK_C , save SK_C kdf pars, sniff



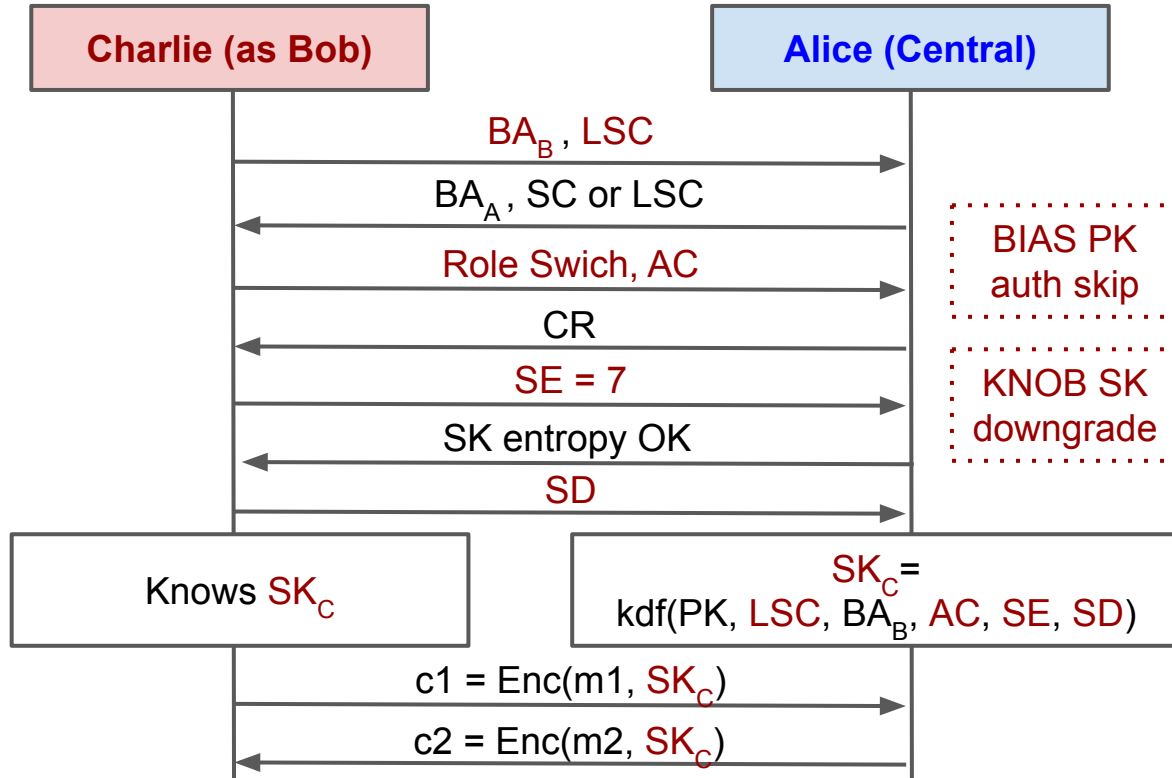
t_2 : Brute force SK_C and break s_{t_1}, \dots, s_{t_2} (break FoS)

- SK_C has 56 bits of entropy (SE = 7)
 - 2^{55} trials on average (other than 2^{55} x sessions)
 - 56 bit sym keys broken since DES ([Deep Crack](#), [COPACOBANA](#))
 - [keylenght.com](#) sets a min of 84 bits (56 bits in 1982)
 - Doable in weeks with a low-cost setup
- SK_C has 8 bits of entropy (SE = 1)
 - Doable in real time (even with pen and paper)

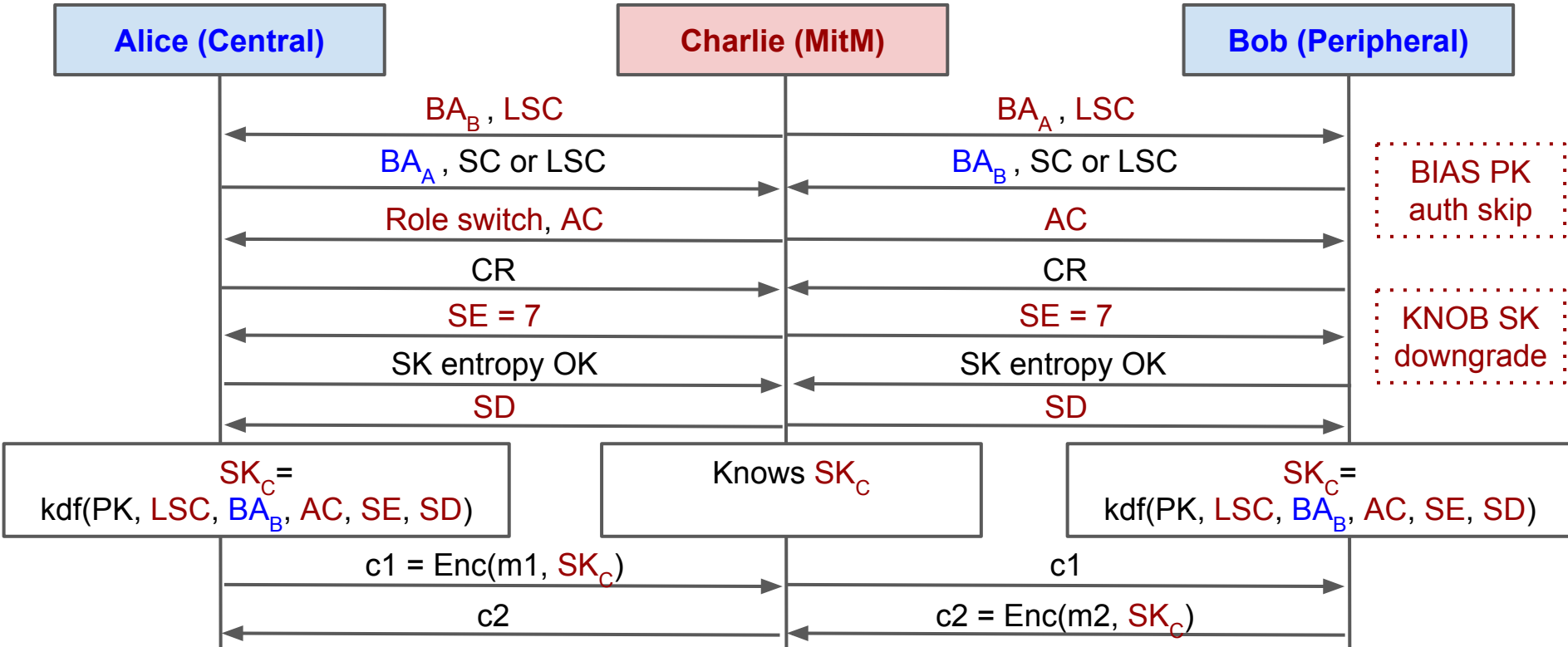
t_3 : Re-force SK_C and break s_{t_3}, s_{t_4}, \dots (break FuS)



t_3 : Re-force SK_C and break s_{t_3}, s_{t_4}, \dots (break FuS)



t_3 : Re-force SK_C and break s_{t_3}, s_{t_4}, \dots (break FuS)



Six BLUFFS Attacks Labels

A1: Spoofing a LSC Central (t_3)

A2: Spoofing a LSC Peripheral (t_3)

A3: MitM LSC victims (t_1, t_3)

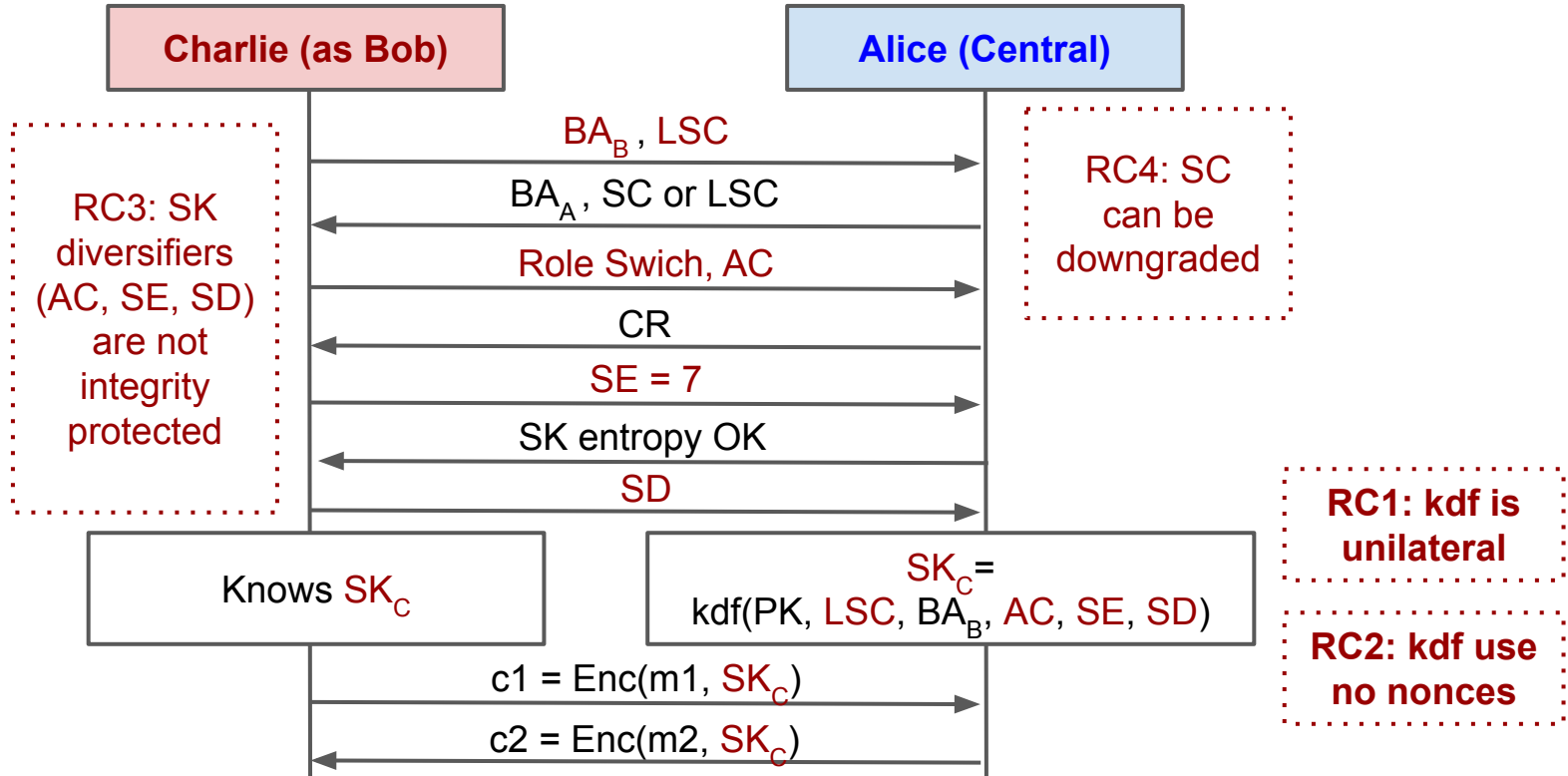
A4: Spoofing a SC Central (t_3)

A5: Spoofing a SC Peripheral (t_3)

A6: MitM SC victims (t_1, t_3)

BLUFFS Vulnerabilities

BLUFFS Attacks have four Root Causes (RC)



BLUFFS Attacks Summary and Root Causes (Vulns)

BLUFFS attack	RC1	RC2	RC3	RC4
A1: Spoofing a LSC Central	✓	✓	✓	✗
A2: Spoofing a LSC Peripheral	✓	✓	✓	✗
A3: MitM LSC victims	✓	✓	✓	✗
A4: Spoofing a SC Central	✓	✓	✓	✓
A5: Spoofing a SC Peripheral	✓	✓	✓	✓
A6: MitM SC victims	✓	✓	✓	✓

BLUFFS Evaluation

BLUFFS Attacks Exploiting 18 devices (17 chips)

Chip	Device(s)	BTv	A1	A2	A3	A4	A5	A6
<i>LSC Victims</i>								
Bestechnic BES2300	Pixel Buds A-Series ³	5.2	✓	✓	✓	✓	✓	✓
Apple H1	AirPods Pro	5.0	✓	✓	✓	✓	✓	✓
Cypress CYW20721	Jaybird Vista	5.0	✓	✓	✓	✓	✓	✓
CSR/Qualcomm BC57H687C-GITM-E4	Bose SoundLink ^{1,2}	4.2	✓	✓	✓	✓	✓	✓
Intel Wireless 7265 (rev 59)	Thinkpad X1 3rd gen	4.2	✓	✓	✓	✓	✓	✓
CSR n/a	Logitech BOOM 3 ¹	4.2	✓	×	✓	✓	×	✓
<i>SC Victims</i>								
Infineon CYW20819	CYW920819EVB-02	5.0	✓	✓	✓	✓	✓	✓
Cypress CYW40707	Logitech MEGABLAST	4.2	✓	✓	✓	✓	✓	✓
Qualcomm Snapdragon 865	Mi 10T ⁴	5.2	✓	✓	✓	×	×	×
Apple/USI 339S00761	iPhones 12 ⁴ , 13 ⁴	5.2	✓	✓	✓	×	×	×
Intel AX201	Portege X30-C ⁴	5.2	✓	✓	✓	×	×	×
Broadcom BCM4389	Pixel 6 ⁴	5.2	✓	✓	✓	×	×	×
Intel 9460/9560	Latitude 5400 ⁴	5.0	✓	✓	✓	×	×	×
Qualcomm Snapdragon 835	Pixel 2 ⁴	5.0	✓	✓	✓	×	×	×
Murata 339S00199	iPhone 7 ⁴	4.2	✓	✓	✓	×	×	×
Qualcomm Snapdragon 821	Pixel XL ⁴	4.2	✓	✓	✓	×	×	×
Qualcomm Snapdragon 410	Galaxy J5 ⁴	4.1	✓	✓	✓	×	×	×

BLUFFS Attacks Exploiting 18 devices (17 chips)

- *LSC Victims*

- All vulnerable
- Except Logitech BOOM 3 against A2, A5 (require Central auth)
- Google Pixel Buds A-Series accept SE = 1 (no KNOB patch)

- *SC Victims*

- All vulnerable if other victim supports LSC
- Eighth devices are not vulnerable to A4, A5, A6 (enforce SC btw pairing and session establishment)

BLUFFS Impact Billions of BT Devices

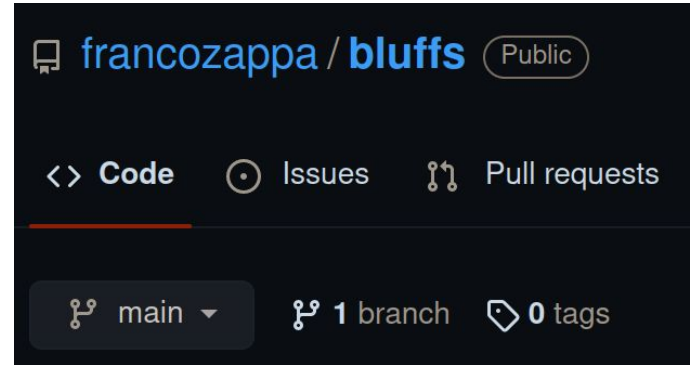
- *Devices*: laptops, smartphones, tablets, headsets, cars, ...
- *OSes*: iOS, Android, Linux, Windows, ...
- *Software*: BlueZ, Gabeldorsche, Bluedroid, proprietary, ...
- *Hardware*: Intel, Broadcom, Logitech, Infineon, Qualcomm, Apple, Microsoft, CSR, ...
- *BT versions*: 5.2, 5.1, 5.0, 4.2, 4.1, ...
- **One BT spec vulnerability → Billions of exploitable devices**

BLUFFS Toolkit

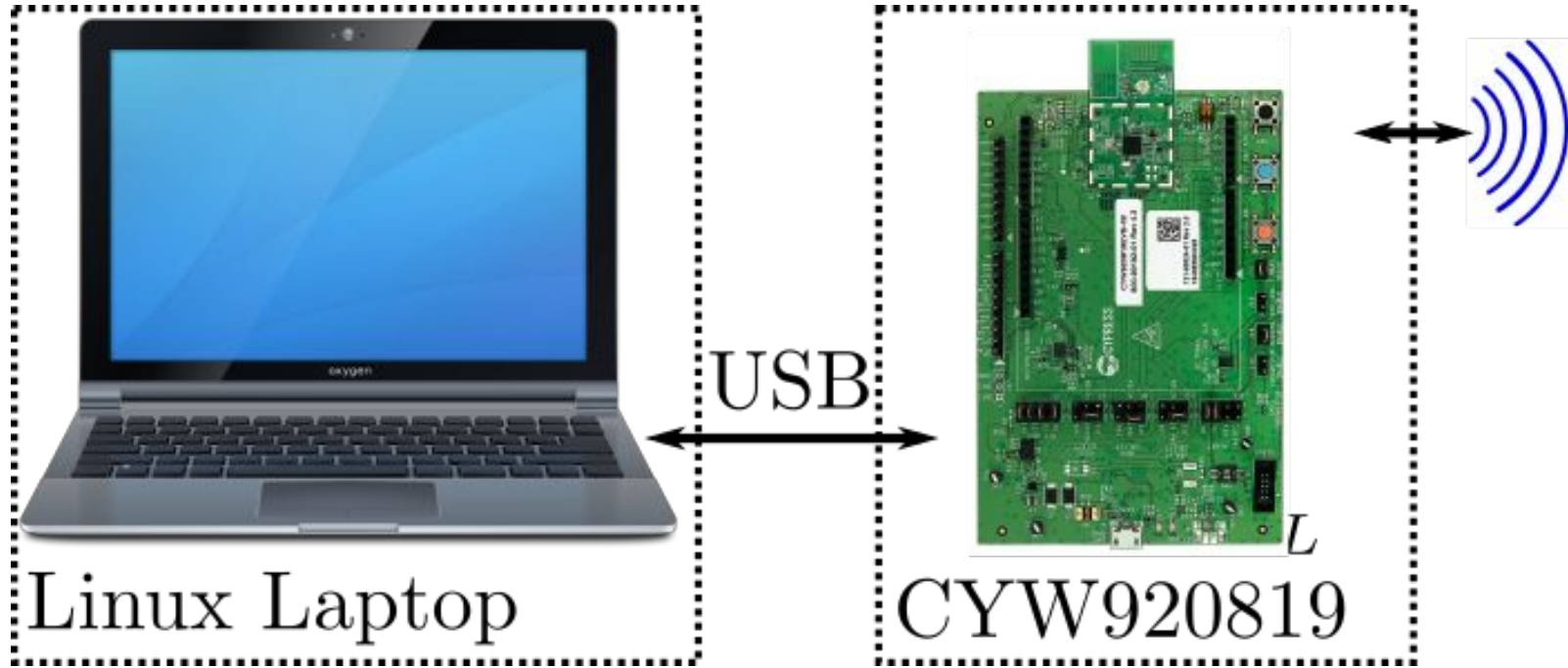
BLUFFS Toolkit

- Features

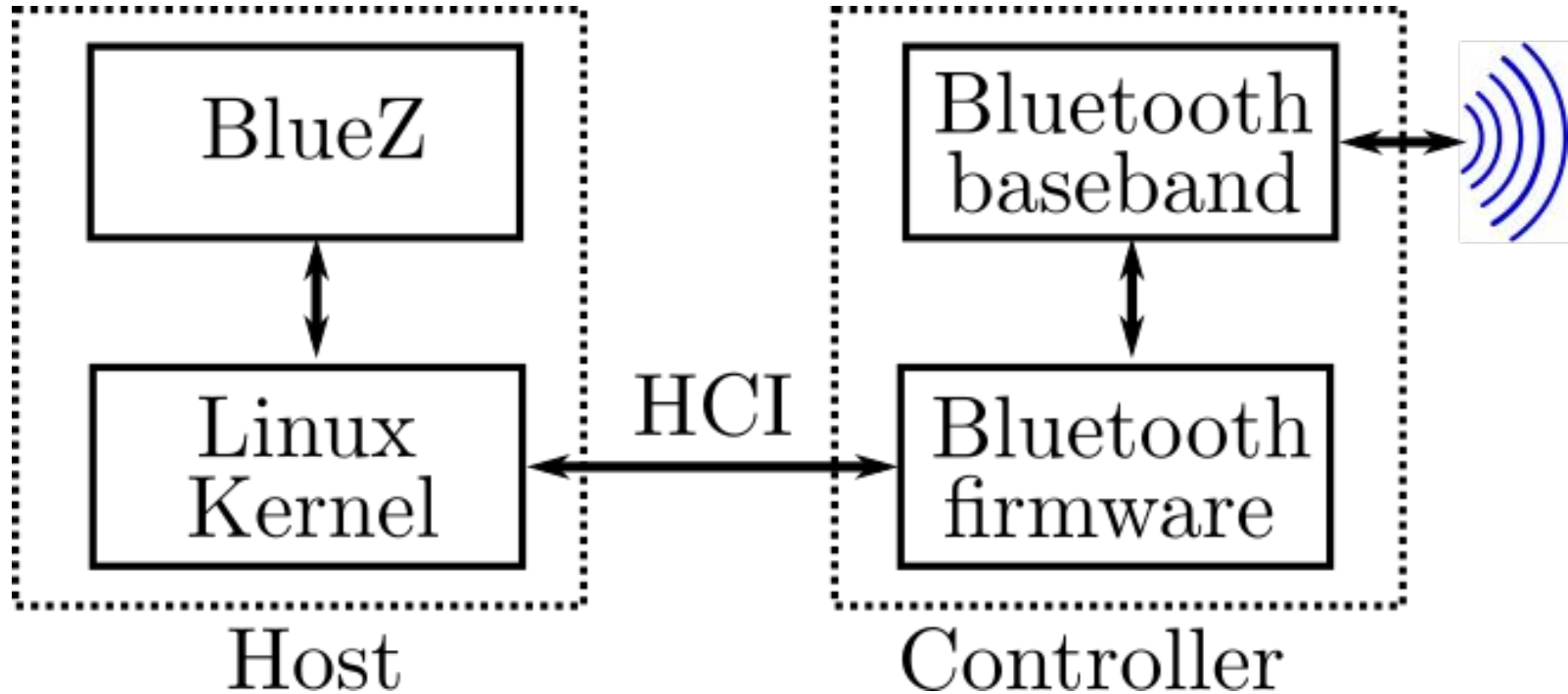
- Open source on [Github](#)
- Perform the attacks ([device](#))
- Detect attacks from a pcap file ([checker](#))
- Sample traffic files ([pcap](#))
- Tamper with SK derivation (**LSC**, **AC**, **SE**, **SD**)
- Monitor SK across sessions (**SK_c**)
- Low cost (Linux laptop, CYW20819 devboard)



Attack Device



Attack Device (2)



Attack Device (3)

- Linux laptop
 - BT Host
 - Parse diagnostic messages (custom Linux kernel)
- Cypress CYW20819 devboard
 - BT Controller (session establishment)
 - Sends diagnostic messages to Host (LMP packets)
 - Reverse engineered BT firmware (luckily we have symbols)
 - BT firmware patched at runtime via [Internalblue](#)

Attack Device ARM Patches ([ref](#))

Name	Description	Patched function	Addr
<i>man_ac</i>	Manip. <i>AC</i>	txAuRand	AEB8C
<i>man_cr</i>	Manip. LSC <i>CR</i>	txSres	AEDC8
<i>man_sd</i>	Manip. <i>SD</i>	txStartEncryptReq	AE4B4
<i>rea_sk</i>	Read <i>SK</i> value	txStartEncryptReq	AE5B4
<i>rea_skec</i>	Read Central <i>SE</i>	txStartEncryptReq	AE5B4
<i>rea_skep</i>	Read Perip. <i>SE</i>	procStartEncryptReq	AE70C
<i>rs_nop</i>	No Perip. role sw.	handleLmpSwitchReq	A643C

man_sd.s ARM patch ([patches](#))

```
1    @LSC SD (EN_RAND) to zero
2    and r0, r0, #0x0
3    str.w r0, [r4, #0x78]
4    str.w r0, [r4, #0x7C]
5    str.w r0, [r4, #0x80]
6    str.w r0, [r4, #0x84]
7
8    @Execute missed instruction
9    ldrb.w r0, [r4, #0x57]
10
11   @Jump to trigger_addr + 5
12   b 0xAE4B9
```

Attack Device Test the BLUFFS Attacks

- Steps

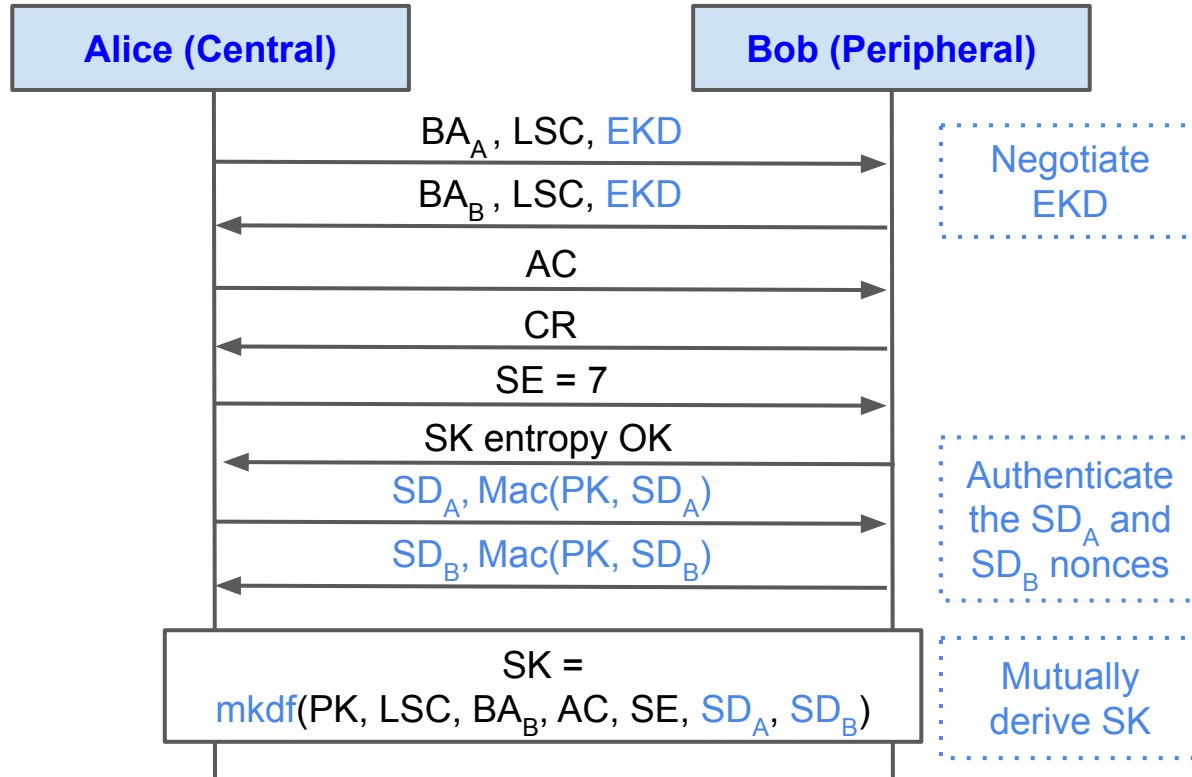
- Connect the board to the laptop
- Activate diagnostic mode (LMP redirection)
- Run Wireshark on the Bluetooth interface
- Pair victim device with attack device and disconnect them
- Run [bluffs.py](#)
- Connect the devices, observe SK_C , disconnect
- Connect the devices, **reobserve SK_C** , disconnect
- **Repeat last step until you are satisfied**

BLUFFS Countermeasures

LSC Enhanced Key Derivation (EKD)

- Features
 - Design level fix
 - Authenticated and mutual SK derivation
 - Backward compliant with LSC kdf
 - Minimal overheads (latency, throughput, ...)
 - Tested at the protocol-level

LSC Enhanced Key Derivation (EKD)



Implementation Level Mitigations

- SC enforcement
 - Enforce SC mode between pairing and session establishment
 - **Cons: protects only SC to SC sessions**
- LSC SD cache
 - Add a cache of seen SDs and check for duplicates
 - **Cons: the attacker can poison the SD cache**
- LSC Central authentication
 - Require mutual authentication before the LSC SK kdf
 - **Cons: fixes a different vulnerability**



Conclusion and Q&A

- First study on BT FoS and FuS ([paper](#), [slides](#))
- Uncover 2 FoS/FuS vulns in BC SK derivation
- Develop 6 BLUFFS attacks breaking BC sessions' FoS/FuS
- Exploit 18 popular devices (Intel, Broadcom, Apple, Google, Microsoft, CSR, Logitech, Infineon, Bose, Dell, Xiaomi, ...)
- Fix the attacks with a compliant and practical protocol
- Report critical findings to BT SIG, got [CVE-2023-24023](#)
- Release [BLUFFS toolkit](#) to test the attacks and BC FoS/FuS

Backup

BLUFFS Lessons Learned

- Pay more attention at issues exploitable across sessions
- Define Bluetooth's **FoS** and **FuS** in the standard
- Develop open source **BC** firmware (Controller)

BT SIG response

BT SIG jargon about BLUFFS vs **reality** [[ref](#)]

- BLUFFS affect BT specifications 4.2 through 5.4
 - **BT versions lower than 4.2 are also affected**
- The researchers identified that a MITM attacker [...]
 - **I also identified spoofing attackers (Central, Peripheral)**
- If a reduced encryption key length can be negotiated
 - **Key length (entropy) can be always reduced**
- Brute forcing of a 7-octet key is not anticipated to be possible in real-time during a session
 - **There is no need to do it in real-time (I bet that someone can)**

BT SIG jargon about BLUFFS vs **reality** [[ref](#)]

- For this attack to be successful, an attacking device needs to be within wireless range of two vulnerable Bluetooth devices
 - **These attacks (MitM and spoofing)**
 - **A spoofing attack requires one victim device in range**
- Implementations are advised to reject service-level connections on an encrypted baseband link with key strengths below 7 octets
 - **BT SIG does not want to fix the standard and re-recommends the KNOB attacks patch from 2019**

BT SIG jargon about BLUFFS vs **reality** [[ref](#)]

- Implementations should also track that a link key was established using BR/EDR Secure Connections in the Bluetooth Security Database [...]
 - **Again, BT SIG does not want to fix the standard and LSC devices and SC devices connecting with LSC ones will be vulnerable forever**
- **BLUFFS Vulnerability (title)**
 - **BLUFFS attacks exploit four vulnerabilities**

Analogies

BLUFFS Attacks Password Analogy

- Victim
 - Set a strong password to login into a service (bank, govt, ...)
 - Required to change the strong password every week
 - Week 2 password useless during weeks 1 and 3 (FoS, FuS)
- Attacker
 - Force the victim to set a **weak password** P_w
 - Guess P_w
 - Trick the victim into resetting P_w every week
 - Login as the victim forever using P_w

Checker

Checker ([ref](#))

- Features

- Object-oriented LMP packet parser ([parser.py](#))
- Implement LSC crypto ([e3](#), [e1](#), [es](#), [h](#), [kdf](#), ...)
- Analyze sessions and detect the attacks ([analyzer.py](#))
- Written in Python 3 using test driven development (TDD)
- Requires [pyshark](#), [BitVector.py](#), and [bitstring](#)

Checker Parser ([ref](#))

LMP packet	Opcode	Description
LMP_host_connection_req	51	Start a session
LMP_detach	7	Abort a session
LMP_encryption_key_size_req	16	Propose <i>SE</i>
LMP_accepted (<i>SE</i>)	3 (16)	Confirm <i>SE</i>
LMP_au_rand	11	Send <i>AC</i>
LMP_sres	12	Send <i>CR</i>
LMP_start_encryption_req	17	Send <i>SD</i>
LMP_accepted (<i>SD</i>)	3 (17)	Accept <i>SD</i>
LMP_not_accepted (<i>AC</i>)	4 (11)	Reject <i>AC</i>

Checker Analyzer ([ref](#))

- Collect sessions packets
- Gen a report per session
- report has SE, AC, SD, SK
- Check if reports have the same SK (EXP_SK)

```
def gen_analysis(PCAP, LK, EXP_SK, BTADD_P):
    """Generate list of sessions and reports"""
    sessions = gen_sessions(PCAP)
    reports = []
    for session in sessions:
        report = gen_report(session, LK, BTADD_P)
        reports.append(report)
    i = 1
    for report in reports:
        print(f"## Begin session: {i}")
        if "keysize" in report:
            print(f"keys: {report['keysize']}")
        if "enrand" in report:
            print(f"enr: {report['enrand'].hex()}")
        if "aurand" in report:
            print(f"aur: {report['aurand'].hex()}")
        if "sk" in report:
            print(f"sk ses: {report['sk'].hex()}")
            # NOTE: check constant SK
            if report["aurand"] == BA_16_ZEROS
                and report["enrand"] == BA_16_ZEROS:
                print(f"sk exp: {EXP_SK.hex()}")
                assert report["sk"] == EXP_SK
        print(f"## End session: {i}\n")
        i += 1
```