Thermal Covert Channel in Bluetooth Low Energy Networks

Timothy Claeys, Franck Rousseau, Boris Simunovic, Bernard Tourancheau

Thermal Covert Channel in Bluetooth Low Energy Networks

Timothy Claeys
Univ. Grenoble Alpes, CNRS
Grenoble Institute of Technology, LIG
Grenoble, France
timothy.claeys@univ-grenoble-alpes.fr

Franck Rousseau
Univ. Grenoble Alpes, CNRS
Grenoble Institute of Technology, LIG
Grenoble, France
franck.rousseau@univ-grenoble-alpes.fr

Boris Simunovic
National Cybersecurity Agency of France (ANSSI)
Grenoble INP, Esisar
Paris, France
boris.simunovic@etu.esisar.grenoble-inp.fr

Bernard Tourancheau
Univ. Grenoble Alpes, CNRS
Grenoble Institute of Technology, LIG
Grenoble, France
bernard.tourancheau@univ-grenoble-alpes.fr

ABSTRACT

This paper investigates thermal covert channels in the context of the Internet-of-Things. More, precisely we demonstrate such a channel on BLE-enabled devices. We explain how the core design principles of BLE link-layer protocol, combined with well-timed intensive CPU calculations can be leveraged to mount a thermal covert channel between two devices.

We implement the attack on three different hardware architectures, a Raspberry Pi 3B, a Motorola X 2014 and an iPhone 5s, and analyze the performance of the channel. We show that we have a similar throughput as previous comparable work, but we greatly improve the range of the channel.

CCS CONCEPTS

• **Security and privacy → Mobile and wireless security**.

KEYWORDS

wireless security, covert channels, Internet-of-Things

ACM Reference Format:
Timothy Claeys, Franck Rousseau, Boris Simunovic, and Bernard Tourancheau. 2019. Thermal Covert Channel in Bluetooth Low Energy Networks. In *12th ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec '19), May 15–17, 2019, Miami, FL, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3317549.3319730

1 INTRODUCTION

The emergence of the Internet-of-Things (IoT) has surrounded us with devices equipped with radio's for wireless communication. The power and computational constraints of these ubiquitous devices have motivated the research community and industry to design energy-efficient protocols. A prime example of such a low-power protocol is Bluetooth Low Energy (BLE) [18]. Since its introduction in the Bluetooth Core specification 4.0 [2], BLE has been implemented on a plethora of devices. Almost all modern smartphones support BLE, making it a very attractive protocol for typical IoT applications such as wireless headphones, smart watches and home automation. BLE allows constrained devices to maintain long-lived connections to periodically send updates to associated devices, e.g. a smartwatch sending heart rate information to a smartphone.

In this paper we investigate the design of a thermal covert channel on BLE-enabled devices. Covert channels allow two colluding devices to secretly exchange data. This can be used to leak sensitive information such as passwords or encryption keys without provoking any security mechanisms. The attack scenario presents a malicious application that secretly leaks sensitive data from a compromised device by piggybacking the data on the legitimate BLE traffic generated by another application. To this end, the attack leverages the design principles of the BLE link-layer protocol. More precisely, we exploit the tight synchronization constraints of the BLE link-layer to build a covert channel between the compromised *victim* device and a *covert receiver*. The covert channel uses the same principles as the side-channel attack presented in [19]. The main idea of the attack is to use heat emissions of the victim device's CPU, triggered by the malicious application, to influence the operation of the crystal oscillator and indirectly the relative clock skew. The heat of the CPU causes a shift in the frequency of the crystal which translates in a measurable clock skew. In contrast to the work presented in [19], the attacker does not interact directly with the victim. Our covert channel leverages the existing traffic of a legitimate BLE connection to an innocent third device, henceforth known as the *helper* device, to hide the communication channel to the covert receiver. The attacker measures directly the varying clock skew of the victim by passively sniffing the traffic between the victim and the helper. As long as the total clock skew stays within the limits of the link-layer's synchronization constraints the covert channel remains hidden and does not disrupt the legitimate connections.

In Section 2 we provide a background in the design of the BLE link-layer protocol and the time synchronization mechanism. We provide an extensive overview on the difficulty of time synchronization and how it is tackled in BLE. In Section 3 we discuss related work on thermal covert channels. Section 4 presents a possible attack scenario detailing the operation of the covert channel. Section 5 explains the communication protocol used to exchange data

WiSec '19, May 15–17, 2019, Miami, FL, USA

ACM ISBN 978-1-4503-6726-4/19/05…$15.00
https://doi.org/10.1145/3317549.3319730

between the victim and the covert receiver. We discuss the requirements to establish a covert channel and provide insight on how we can improve the reliability and throughput of the channel. In Section 6 we present an implementation of the covert channel on three different hardware platforms: a Raspberry Pi 3B, a Motorola X 2014 and an iPhone 5s, respectively running Arch Linux, Lineage OS 14.1 (equivalent to Android Nougat 7.1.2) and iOS 11.4.1. We use a simple application that sets up a persistent BLE connection to a helper device, i.e. a smart light bulb. At the same time a malicious application schedules threads of cryptographic calculations in the background to heat up the crystal oscillator and modulate bits under the form of a changing clock skew. The malicious application does not require any special privileges. As covert receiver we use an Ubertooth One [10], a simple off-the-shelf Bluetooth sniffer, to passively measure the varying time intervals between the packets emitted from the victim to the smart light bulb. On the Raspberry Pi platform we obtain a throughput of 38 bits per hour and 45 bits per hour with additional cooling. The throughput on the Motorola is limited to 8 bits per hour, while on the iPhone we obtain 32 bits per hour. The differences in throughput can be attributed to the layout of the hardware and the use of temperature compensated crystals (TCXO). Finally we compare the performance of our channel to previous work.

2 BACKGROUND

In this section we provide the necessary background information on BLE and clock drift in crystal oscillators.

2.1 Bluetooth Low Energy

BLE was first introduced in the Bluetooth Core specification v4.0. In contrast to classic Bluetooth which can transfer a lot of data, BLE is optimized for low-power applications that only periodically need to communicate. BLE allows devices to go into a sleep mode between data transmissions, effectively lowering the overall power consumption of the entire protocol. The Bluetooth communication stack can roughly be divided in three parts [11]. At the bottom we can find the controller part. The controller contains the lowest two layers of the Bluetooth stack: the physical layer and link-layer. The controller is typically implemented in the firmware of the Bluetooth radio. The host encompasses the higher levels of the Bluetooth stack, such as the logical link control, attribute protocol, generic access profile, etc…The host is often implemented as a part of the operating system of the device that uses Bluetooth networking. On top of the host we can find the Bluetooth applications. These are traditionally implemented in the user space. In this paper, we are only interested in the the link-layer or medium access layer (MAC) of the stack. This layer describes and implements the algorithms that regulate access to the transmission medium. Similar to other low energy protocols that target IoT and embedded systems [1], BLE devices share a global network clock to synchronize the devices. The BLE devices follow a strict schedule that combined with the global clock informs the devices when they are allowed to transmit data or could expect an incoming packet. The rest of the time the devices can go into sleep mode. Additionally, BLE uses channel hopping. It can use up to 40 different frequency channels to communicate.

Although in the most recent update of the Bluetooth Core specification [3], version 5.0, changes were introduced to allow for multi-hop networking, BLE is most frequently used to establish a peer-to-peer connection between two devices. The devices can have one of two distinct roles: the slave or master role. In a typical use case the slave is a very simple device, e.g. a temperature sensor or light bulb, that generates some application data. The master is a more powerful device such as smartphone. To establish a connection a master device listens for advertisement packets, periodically emitted by the slave during an *advertisement event*. When a master device has captured an advertisement packet it can start a negotiation phase by responding with a *connection request packet*. During the connection setup the master and slave negotiate the parameters for the connection. The connection request packet of the master device contains among other a *connection interval* and a pseudo-random sequence. The connection interval indicates how often the master and slave are going to wake-up to exchange data, known as a *connection event*, and the pseudo-random sequence describes the order in which the frequency hopping takes place. Values for the connection interval range from 7.5 ms to 4 s and are a multiple of 1.25 ms [3]. The start of a connection event is called the *anchor point*. The master device initiates the connection event by sending a packet to the slave, see Figure 1. This packet can contain application data but more often it simply acts as a poll request, asking the slave to respond with its application data. The slave answers either with its data or with an empty packet in case it has no data to transmit. Both devices keep following this schedule until the master disconnects.

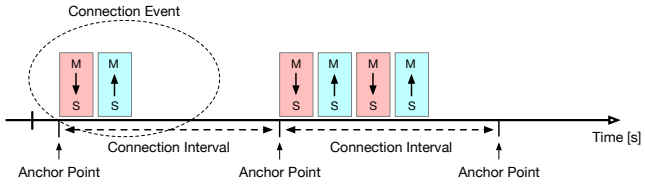

Figure 1: Bluetooth Low Energy in connected mode. During a connection event multiple packets can be exchanged.

The master and slave can also negotiate a *slave latency* parameter. This parameter defines the number of consecutive connection events that the slave device is not required to listen for the master's packet at the start of a connection event.

2.2 Challenges of time synchronization

To save energy BLE devices enter in a sleep mode between two connection intervals. The devices use an internal clock system driven by a low-power crystal oscillator to keep track of time. An interrupt wakes up the master and the slave when a new anchor point is reached. It is obvious that the internal clocks of the BLE devices need stay synchronized in order to wake up at the same moment. Maintaining this tight time synchronization is not trivial. Due to impurities in the crystal oscillator, manufacturing imperfections and environmental changes (e.g. temperature changes) the exact frequency of the crystal can differ from the target frequency. The offset between the targeted frequency and the real frequency is

called the clock drift, expressed in parts per million (ppm). Different crystals have different clock drifts. Typical values for the clock drift range between 20 ppm to 40 ppm. A relative clock drift over time between two devices leads to a clock skew. The clock skew is defined as the difference in elapsed time between two devices. Equation 1 describes all the factors that amount to a clock skew over time. The $E(t)$ factor represents the varying drift from external causes, i.e. temperature changes. The $\Delta\epsilon$ factor denotes the difference in frequency shift between two devices (static clock drift) and $\pm e_f$ describes the production spread (caused by imperfections in the manufacturing process) [7].

$$\Delta C_{skew} = \Delta t\left(\frac{1}{1-e_f} - \frac{1}{1+e_f} + E(t) + \Delta\epsilon\right), \quad (1)$$

The Bluetooth Core specification specifies two different maximum clock accuracies. During a connection event or advertising event the devices use the active clock accuracy, with a drift less than or equal to ±50 ppm. When in sleep mode the devices use the sleep clock accuracy, which can have a maximum drift of ±500 ppm. Because of the potential high drift and the resulting clock skew there is an uncertainty in the slave device of the exact timing of the master's anchor point. Therefore, the slave is required to resynchronize to the master's anchor point at each connection event to avoid drifting too much. When the slave receives a packet from the master, the slave updates its anchor point [3] and always responds with a packet unless the slave latency parameter is not 0. To ensure that the slave wakes up in time to receive the first packet of the master during an connection event, it estimates the next anchor point, taking into account the possible clock drift. During the connection setup the master can indicate its sleep clock accuracy. The slave can use the master sleep clock accuracy and its own sleep clock accuracy to better estimate the position of the next anchor point. The slave calculates a window around the expected anchor point. The slave will start listening for a packet from the master a full window widening value before the expected anchor point, see Figure 2.

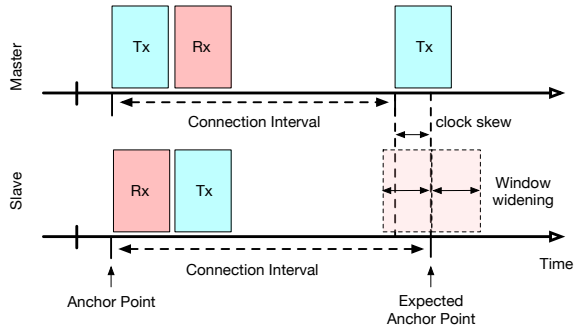

Figure 2: Clock skew between master and slave. Slave uses window widening technique to compensate the clock skew.

The formula used to calculate the window widening is shown in Equation 2.

$$\delta_{window} = \Delta t\left[(D_m + D_s)/1 \times 10^6\right] \quad (2)$$

The Δt parameter indicates the elapsed time since the last anchor point (and resynchronization of the slave to the master). The values D_m and D_s are the master's and slave's clock inaccuracies, expressed in parts per million (ppm), respectively.

3 RELATED WORK

There exists a large variety in covert channel attacks, targeting different types of hardware. However, in this paper, we focus on thermal covert channels to exfiltrate sensitive information.

Our approach is based on the work presented in [19]. This work discusses a side-channel attack to discover hidden Tor services. Murdoch shows that changes in clock skew resulting from only modest changes in temperature, can be remotely detected through network packet timestamps, even over tens of router hops. The primary contribution demonstrates an attack whereby CPU load induced through one communication channel affects clock skew measured remotely. The attacker uses the TCP timestamp option to measure the clock skew. This observation is then used to detect hidden Tor services. Murdoch is able to link a pseudonym to a real identity, even against a system that ensures perfect non-interference.

Guri et. al. [12] present Bitwhisper, a thermal covert channel between air-gapped adjacent computers. It exploits the thermal radiation emitted by one computer, operating within permissible heat boundaries, to deliver information to a neighboring computer, equipped with standard heat sensors. Guri et. al. also discuss signal modulation and communication protocols, showing how BitWhisper can be used for the exchange of data between two computers in close proximity. The range of the covert channel is limited to 40 cm and the throughput of the channel amounts to 1 to 8 bits per hour, a rate which makes it possible to infiltrate brief commands and exfiltrate small amount of data (e.g., passwords) over the covert channel a covert communication channel.

The work presented in [17], demonstrates the feasibility of thermal covert channels on multicore platforms. They show that even seemingly strong isolation techniques based on dedicated cores can be circumvented through the use of thermal channels. Specifically, they show that the temperature of a single processor core, measured by a neighboring core, can be used both as a side channel as well as a covert communication channel even when the system implements strong spatial and temporal partitioning. They test their hypothesis on an Intel Xeon server platform. The thermal covert channel achieves a throughput 12.5 bps. Additionally the authors show that the thermal side-channel can be used to profile applications running on the neighboring processor cores.

4 ATTACK SCENARIO

In a typical attack scenario, a user gets tricked into installing a malicious application. The malicious application could be impersonating a legitimate application, or the legitimate application itself was compromised due to a supply chain attack [20]. The malicious application has access to sensitive data such as passwords or encryption keys (e.g. a password manager). However, the application cannot use any networking functionalities, as the user did not grant the application those privileges. It has no direct way to extract the sensitive data from the victim device. Instead the malicious application modulates the sensitive data on top of the packets exchanged

by long-lived BLE connections of legitimate applications. Examples of such long-lived connections are: a smart watch being connected to a smartphone or a smartphone connected to wireless headphones or smart loudspeakers.

Figure 3 depicts the attack scenario. There are three distinct entities: the victim device, a neutral helper device and the covert receiver which is fully controlled by the attacker. By scheduling well-timed CPU intensive calculations, the malicious application can control the heat emission of the CPU. These emissions directly influence the frequency of crystal oscillator. The malicious application uses the impact of the $E(t)$ in Equation 1 to cause a varying, measurable clock skew. Bits can be encoded under the form of these clock skew variations.

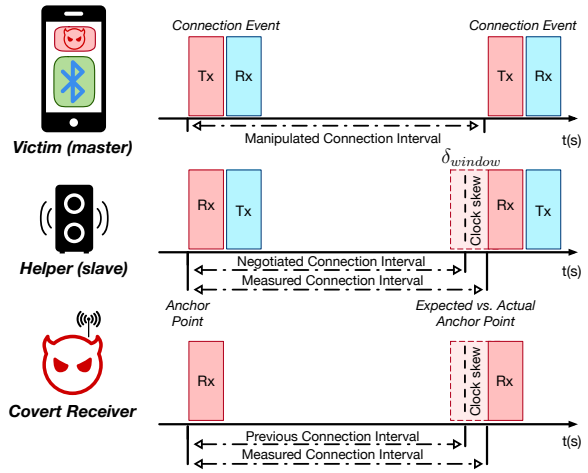

Figure 3: Timeline of the attack. The covert receiver infers the induced clock skew, caused by the CPU heat, by measuring the changing connection interval between the consecutive anchor points (icon source: [25])

The synchronization constraints in the BLE protocol force the master and slave to exchange packets every connection interval. This provides the covert receiver with an easy and stealthy way to track the relative clock skew between the two devices. The covert receiver does not need to inject any traffic in the network or ask for explicit timestamps to estimate the clock skew. It suffices to passively sniff the connection and measure the time between two anchor points. At the start of each connection interval (at the anchor point) the master will poll the slave for any application data. If the clock frequency changed sufficiently since the last connection event, the connection interval will have changed noticeably. The covert receiver tracks these changes and decodes them. The covert receiver does not need to know the contents of the BLE-packets. This means that the covert channel can also be established when link-layer security encrypts the content of the packets.

The victim should always have the role of master in the BLE connection as this forces the helper device to stay synchronized to the victim even if there is a large clock skew. The covert receiver can be positioned anywhere in BLE transmission range and acts as a hidden slave device. There are no physical or logical connections between the victim and the covert receiver. The covert receiver can even be placed in an adjacent room. As long as the impact of $E(t)$ in Equation 1 is sufficiently large with respect to the other terms in the equation the covert receiver can measure with sufficient precision the intervals between the consecutive BLE transmissions.

5 COVERT COMMUNICATION PROTOCOL

In this section we describe the communication protocol between the victim device and the covert receiver. Because the communication channel is unidirectional we allow a for significant overhead for error correction codes as there is no way the covert receiver can ask for the retransmission of data.

5.1 Encoding and Decoding

In our communication protocol we use a simple On-Off Keying (OOK) technique to transmit data. To encode a '1' bit a strong variation in the clock skew is triggered by heating up the oscillator. A '0' bit is encoded as the absence of a varying clock skew. We used several threads of cryptographic calculations to generate a high CPU load. The malicious application must keep the temperature steady for a period of time. This ensures that the heat can propagate throughout the device and reach the oscillator. When the CPU load is released by the malicious application, the device and crystal oscillator cools down and the clock skew returns to its previous value. During this process the covert receiver is measuring the elapsing connection intervals (CI_j) by creating a timestamp (T_i) at the reception of the first packet of the master at the start of an connection event.

$$\mathrm{CI}_j = (T_i - T_{i-1}) \tag{3}$$

By subtracting the subsequent connection intervals, the attacker calculates the clock skew for the connection interval.

$$\Delta C_{skew} = \Delta\mathrm{CI} = \mathrm{CI}_j - \mathrm{CI}_{j-1} \tag{4}$$

The attacker then uses a low-pass filter to extract the trends in the clock skew behavior, $S(t)$. By calculating the derivative, $\frac{dS}{dt}$, over the filtered signal the attacker can detect sudden strong variations in the clock skew. When the absolute value of the clock skew variation is greater than a predefined threshold value V_{th} a '1' bit is detected, otherwise the filtered signal is decoded as a '0' bit.

$$\text{Bit string}(t) = \begin{cases} 1 & |\frac{dS}{dt}| \geq V_{th} \\ 0 & |\frac{dS}{dt}| < V_{th} \end{cases} \tag{5}$$

The threshold value V_{th} is discovered by calculating the standard deviation (σ) over $\frac{dS}{dt}$. A scaling factor α is experimentally derived.

$$V_{th} = \sigma\left(\frac{dS}{dt}\right) * \alpha \tag{6}$$

Each message is prepended with a preamble. The preamble marks the start of a data transmission. We use a preamble of a single '1' bit to denote the start of a transmission. We must choose a preamble starting with bit '1' because only a change in the clock skew can be detected by the covert receiver. A '0' bit is encoded as the absence of a varying clock skew and can therefore not be clearly identified by the decoder.

5.2 Calibration of the channel

Prior to the attack, we calibrate the channel to derive the optimal parameters for the covert channel. The calibration values improve the throughput and accuracy of the channel. Without these values the malicious application needs to guess for how long it has to heat up the device and, similarly, how much time the device needs to cool down. Initially, the base temperature of the device, T_b must be measured. The base temperature corresponds to temperature of the CPU when it is idle. Secondly, the target temperature, T_t, must be defined. Thirdly, we must set the time period, Δt_h, for which the malicious application must hold the CPU temperature at T_t when encoding a binary '1'. Finally, the malicious application needs to know the cool down time, Δt_c. This variable describes the time needed by the system to return to the base temperature T_b and for the clock skew to settle on its original value. Calibration happens offline, once per hardware architecture and once for different base temperatures.

5.3 Error correction and data whitening

To provide reliable transmission the malicious application uses two techniques: error correction codes and data whitening. A possible candidate for the error correction codes are the Bose-Chaudhuri-Hocquenghem codes (BCH) [4]. BCH codes can correct multiple errors, depending on construction of the code word. For example the BCH(63, 31) code adds maximally 31 check-bits to a data payload of 32 bits. This would allow the covert receiver to correct up to 5 faults in the received code word.

Data whitening scrambles the data before transmission to prevent long sequences of 1's or 0's. Because each binary '1' is encoded as an increase in CPU temperature there exists a risk that several consecutive encoded '1' bits could saturate the sensitivity of the oscillator to temperature changes. As remnant heat builds up in the system, the device does not have enough time to cool down. We borrow the data whitening mechanism described in the Bluetooth Core specification [3]. A simple LFSR is used to generate a pseudo-random bit string. This bit string is then xored with the original data to obtain a pseudo-random bit sequence, ready for transmission.

6 EXPERIMENTAL EVALUATION

6.1 Experiment setup

In this section, we discuss our implementation of the attack on three hardware platforms: Raspberry Pi 3B, Motorola X 2014 and an iPhone 5s. All the experiments followed the scenario described in Section 4. The experiments were performed in an office at room temperature. In each experiment the role of the helper device was fulfilled by a simple BLE-enabled light bulb [24]. The covert receiver consisted of an Ubertooth One [10] connected to a computer. The distance between the different devices was approximately 2 m.

6.2 Tracking the clock skew

The Ubertooth is capable of sniffing a specific BLE connection when it can overhear the initial BLE connection setup. This handshake negotiates all the parameters necessary to calculate the next anchor points and to derive the frequency hopping. The Ubertooth firmware registers timestamps at the reception of every packet. The timestamps are generated by the Ubertooth's CPU [21] by reading the current timer value at the reception of radio direct memory access (DMA) interrupt. The timer runs with a period of 50 MHz. The Ubertooth sends the timestamps to the computer. We only keep the timestamps of the first packet emitted by the victim/master device at every connection event. The difference between two consecutive timestamps corresponds to the connection interval of the victim, see Equation 3. The connection interval remains stable when the oscillator operates at a fixed temperature (a constant clock skew between the victim and the helper/covert receiver). In this scenario all the terms in Equation 1 are stable, resulting in a stable clock skew. If the oscillator experiences a fluctuating ambient temperature, the $E(t)$ term causes a varying clock skew and the Ubertooth measures varying connection intervals.

6.3 Victim configuration

As victim device, containing the malicious application, we used three different devices: a Raspberry Pi 3B [9] running Arch Linux, a Motorola X 2014 running Lineage OS 14.1 and an iPhone 5s with iOS 11.4.1. Before the start of the experiment, during the reconnaissance phase, the physical characteristics of the victim device are investigated. We try to register the base temperature of each device, the temperature of a device when idle, the amount of heat that was effectively generated and how fast the heat propagated throughout the device and finally how quickly this heat dissipated. We also verified the size of the clock skew variations due to the increasing or decreasing temperature. Large variations point towards a direct thermal path between the CPU and the oscillator and/or the use of an uncompensated crystal oscillator, while small variations might be caused by the CPU and oscillator being thermally isolated and the use of a TCXO.

6.4 Raspberry Pi 3B

6.4.1 Reconnaissance phase. The Raspberry Pi 3B board uses a quad core 1.2 GHz Broadcom BCM2835 64-bit CPU and a BCM43438 chipset for WiFi and BLE connectivity [6]. Figure 5 shows the positions of the CPU and the wireless chipset with crystal oscillator on the board. The distance between the CPU and the oscillator is approximately 1.5 cm. Both the CPU and BLE chip are attached to the same board, allowing for a direct thermal path, see Figure 6.

After inspection of the board and the datasheet of the BCM43438 we suspect that the board uses an uncompensated crystal oscillator, although verification of this assumption is not possible as the majority of the Raspberry Pi schematics are not publicly available. The board also exposes one internal CPU temperature sensor. In a first instance we analyzed how well the heat of the CPU propagates throughout the board. With a thermocouple [8] we track in real-time the temperature of the oscillator as we vary the load on the CPU. We use this information to derive the values to calibrate the covert channel.

Figure 6 shows the heat spread from the CPU throughout the board. The article [13] discusses the changes in heat propagation between the Raspberry Pi 3B and the Raspberry Pi 3B+. Although the

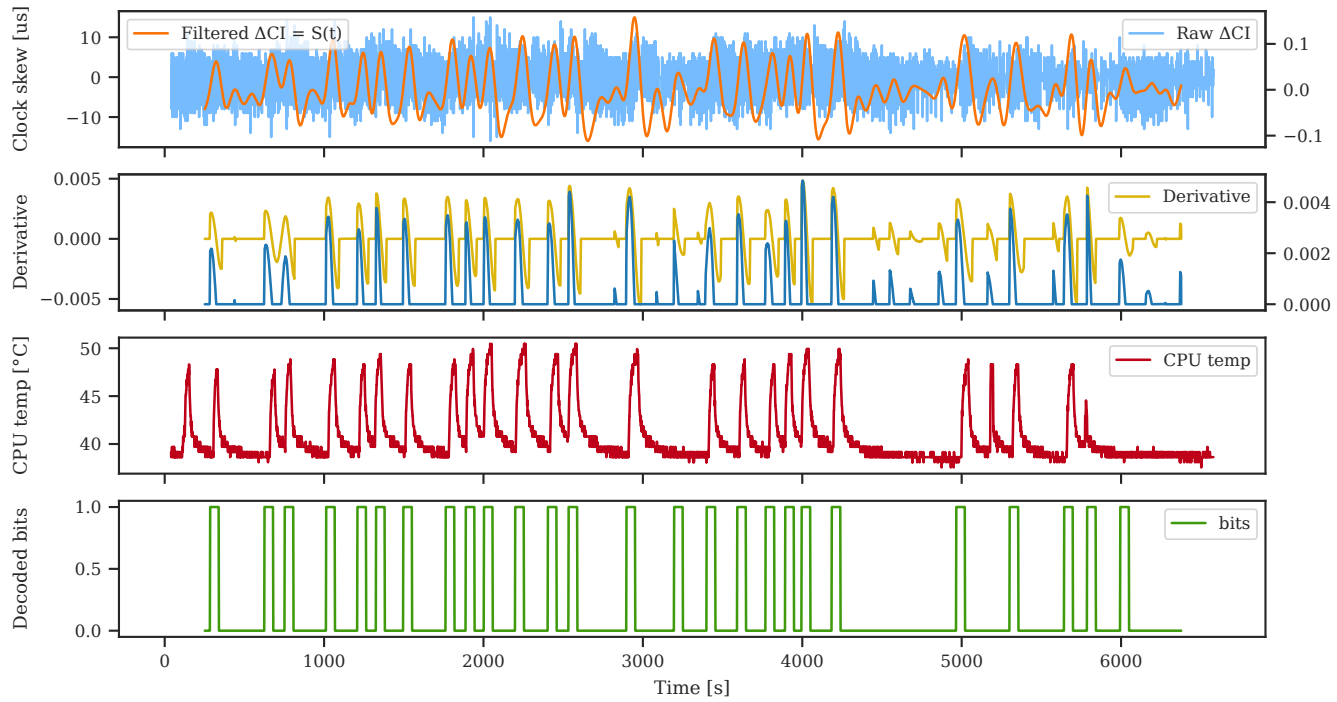

Figure 4: Implementation of the covert channel on the Raspberry PI 3B. From top to bottom: raw and filtered skew, the derivative of the filtered skew, the CPU temperature and the decoded bits.

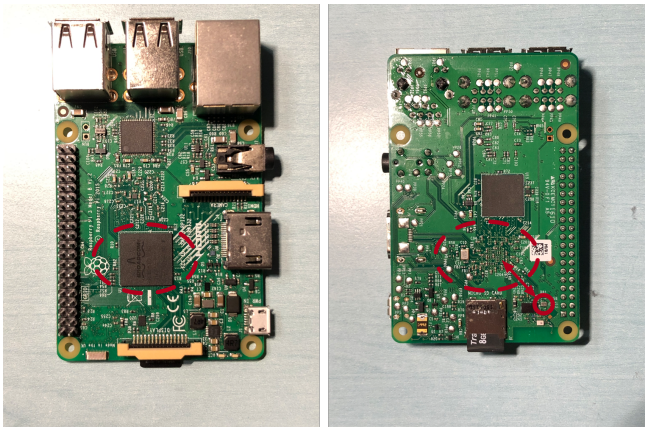

Figure 5: The dashed circle indicates the position of the CPU on the Raspberry Pi's logic board. The full circle shows the crystal oscillator.

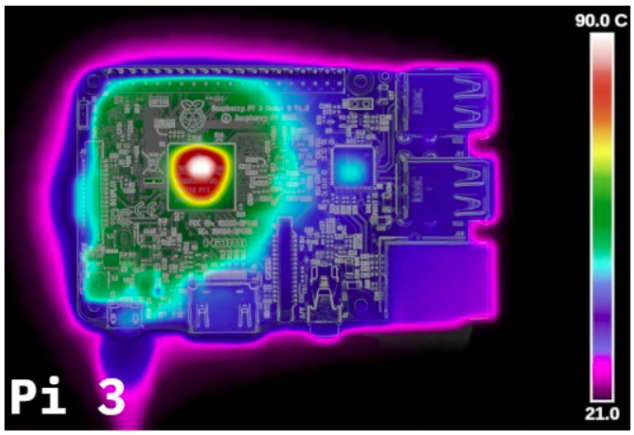

Figure 6: Thermal image of the RPi 3B (source: [13]).

newer model has a better heat spread, the heat propagation is sufficiently wide to reach the oscillator. Measurements with thermocouple [8] indicate 10 °C to 15 °C difference between the temperature measured at the CPU package and the oscillator.

6.4.2 Covert Channel Performance. With the characteristics of the victim known, the malicious software on the Raspberry Pi can set up a the covert channel. Figure 4 shows a covert transmission of approximately 2 h. The raw skew data is directly measured by the Ubertooth and logged by the computer. By filtering the signal we extract the clock skew trend from the original noisy signal. The second graph in Figure 4 shows the derivative of the filtered signal. We also plot the derivative a second time with the negative values set to 0. We observe that the peaks in CPU temperature match well with clock skew variations, shown by the peaks in the derivated signal. Finally we use Equation 5 and Equation 6 to extract the bits,

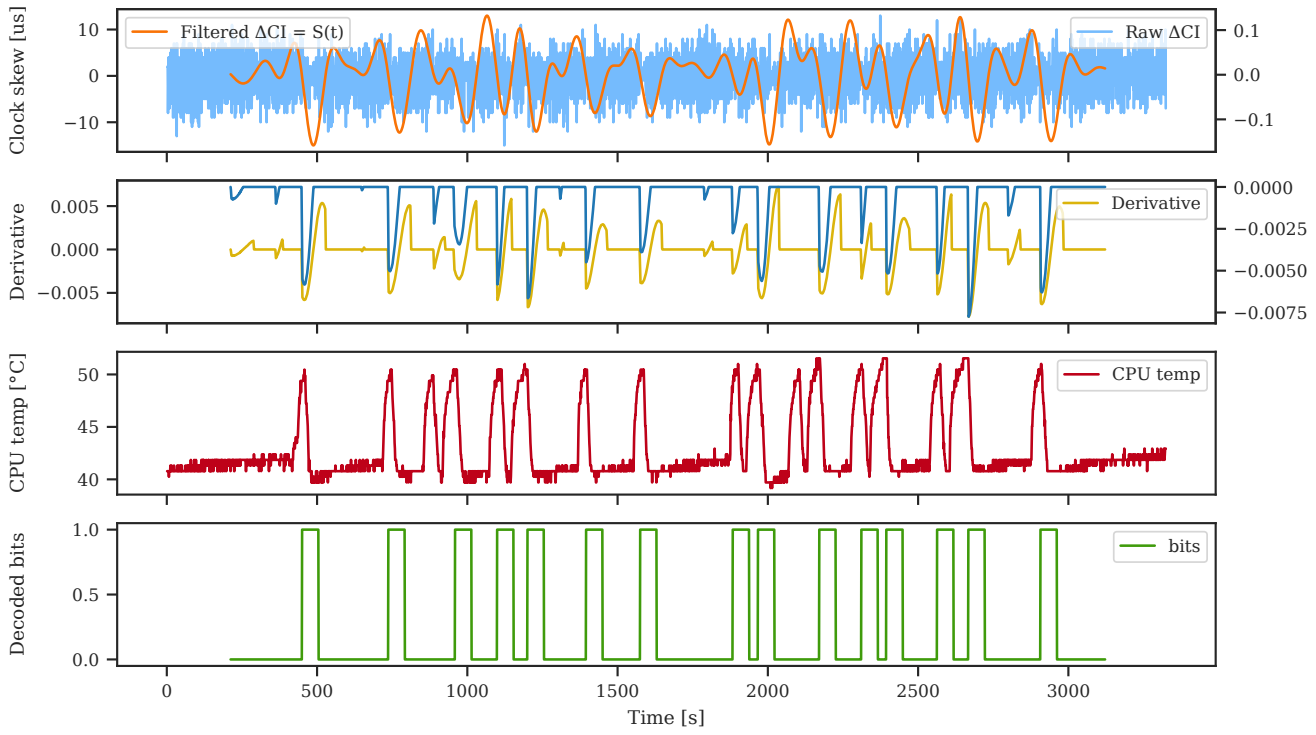

Figure 7: Covert Channel on the cooled Raspberry Pi. The cool down phase is quicker then the heat up phase. We search for strong negative spikes in the filtered clock skew signal instead of positive spikes.

Table 1: Calibration values for the covert channel: RPI-3B

Constant	Value	Comments
T_b	39 °C	Temperature for idle CPU
T_t	47 °C	T_t + 1 for every consecutive '1' bit
Δt_h	10 s	
Δt_c	65 s	Cool down time: $T_t \rightarrow T_b$

shown in the lowest graph of the figure. The throughput of the channel is approximately 38 bits per hour.

To improve the throughput of the channel, we needed to accelerate the cool down phase of the covert channel. In a second experiment we assumed that the malicious software also controlled a fan. This fan could be activated to help cooling the CPU and the oscillator. We used the same base temperature and target temperature, but the cool down time, ΔT_c, was reduced to 30 s. The results are shown in Figure 7. The throughput of the new cooled channel is approximately 45 bits per hour.

6.5 Motorola X 2014

6.5.1 Reconnaissance phase. The Motorola X 2014 uses a Qualcomm Snapdragon 801 8974-AC CPU [22] and a Qualcomm WCN3680 802.11ac Combo Wi-Fi/Bluetooth/FM chipset [23]. The wireless chipset of the phone uses, highly probable, a TCXO. A full teardown of a first generation Motorola X shows the approximate locations of the CPU and the BLE chip [15]. The CPU and BLE chip seem to be far apart. A teardown of the second generation Motorola (used in our experiments) additionally shows metal heatsinks covering the individual chips. This hardware layout would allow to evacuate the majority of the generated heat before it reaches the BLE chip. If the chip additionally uses a TCXO the effects of the CPU heat on the induced clock skew could be negligible. Because we have no access to the actual schematics of the phone, we cannot verify these assumption.

The phone's hardware has 15 different temperature sensors. Lineage OS makes the sensor values readable in the filesystem under `/sys/class/thermal/thermal_zone[1-15]/temp`. Although many of the sensor names are cryptic, some of them can easily be attributed to hardware components of the phone, e.g. `chg_temp` exposes the temperature sensor of the battery. After careful testing the temperature values, returned by the sensors during CPU intensive calculations, we pick a sensor for our experiment which corresponds well to the measured clock skew.

6.5.2 Covert Channel Performance. Compared to the experiment with the Raspberry Pi 3B, the throughput of the covert channel on the phone is much smaller, see Figure 5. The throughput is approximately 6 to 8 bits per hour. Several reasons can be found that explain this performance drop. The heat up time and cool down time are much larger. The slow increase in temperature is probably due to the use of metal heat sinks over the different chips, preventing the heat of the CPU to spread to the other components

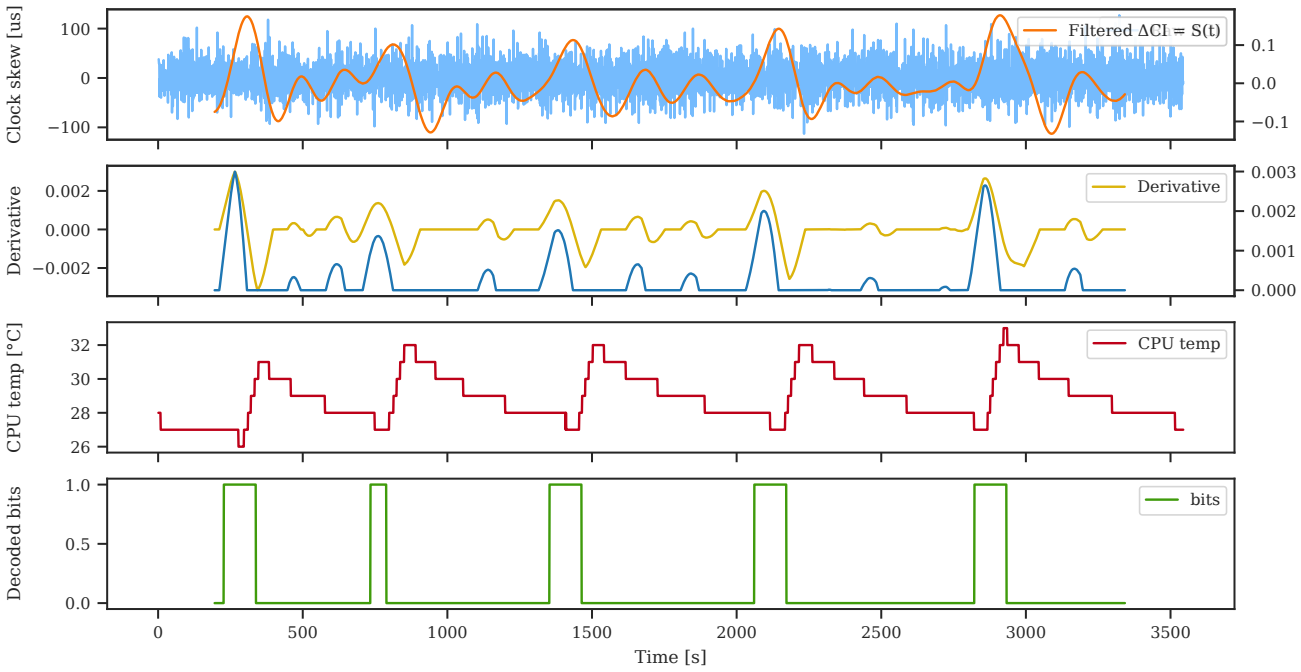

Figure 8: Covert channel on the Motorola X 2014. Compared to the filter clock skew on the Raspberry Pi platform, the clock skew variations are much more suppressed.

Table 2: Calibration values for the covert channel: Motorola

Constant	Value	Comments
T_b	27 °C	Temperature for idle CPU
T_t	32 °C	Target temperature
Δt_h	1 s	
Δt_c	±400 s	Cool down time: $T_t \rightarrow T_b$

Table 3: Calibration values for the covert channel: iPhone 5s

Constant	Value	Comments
T_b	-	No access to temperature sensors
T_t	-	No access to temperature sensors
Δt_h	±60 s	
Δt_c	±70 s	Cool down time: $T_t \rightarrow T_b$

on the motherboard. Once the oscillator has reached it's target temperature we cancel the CPU load. In contrast to the Raspberry Pi the phone's hardware is protected by its exterior casing. This prevents the build up heat from dissipating easily. The usage of the TCXO also prevents strong clock skew variations which make the detection of peaks in the filtered signal harder.

6.6 iPhone 5s

6.6.1 Reconnaissance phase. The iPhone 5s uses Apple's A7 chip, a custom ARMv8 1.3 GHz dual core processor [16]. The WiFi/Bluetooth chipset is based of Broadcom's BCM4334 [5]. The BCM4334 is similarly to the BCM43438 capable of using a TCXO to compensate temperature variations. The iPhone's OS does not expose any internal temperature sensors. A teardown of the iPhone 5s reveals the positions of the A7 and BLE chip [14]. Compared to the Motorola's logic board, the iPhone's BLE chip seems to be in close proximity to the A7 processor.

6.6.2 Covert Channel Performance. The iPhone 5s has a much higher throughput compared to the Motorola. The throughput is approximately 32 bits per hour. The influence of the CPU heat on the oscillator is rapidly noticeable. Additionally, the clock skew returns fast to its base value when we stop heating up the CPU. We can not say with certainty why the iPhone's oscillator is more susceptible to the temperature changes but our guess is that the close vicinity of the BLE chip to the A7 application processor allows for a direct thermal path. The phone's heat sink also seems to be more efficient which allows for a fast heat dissipation. The iPhone's exterior casing is additionally surrounded by a metal band at the sides that functions as a large heatsink.

6.7 Performance discussion

The bandwidth of our covert channel depends on several factors. This includes the time required to heat up or cool down the oscillator to a specific temperature and the error rate in the transmission. Both these parameters in turn depend on the hardware architecture of the victim and the distance between the heat source and the crystal

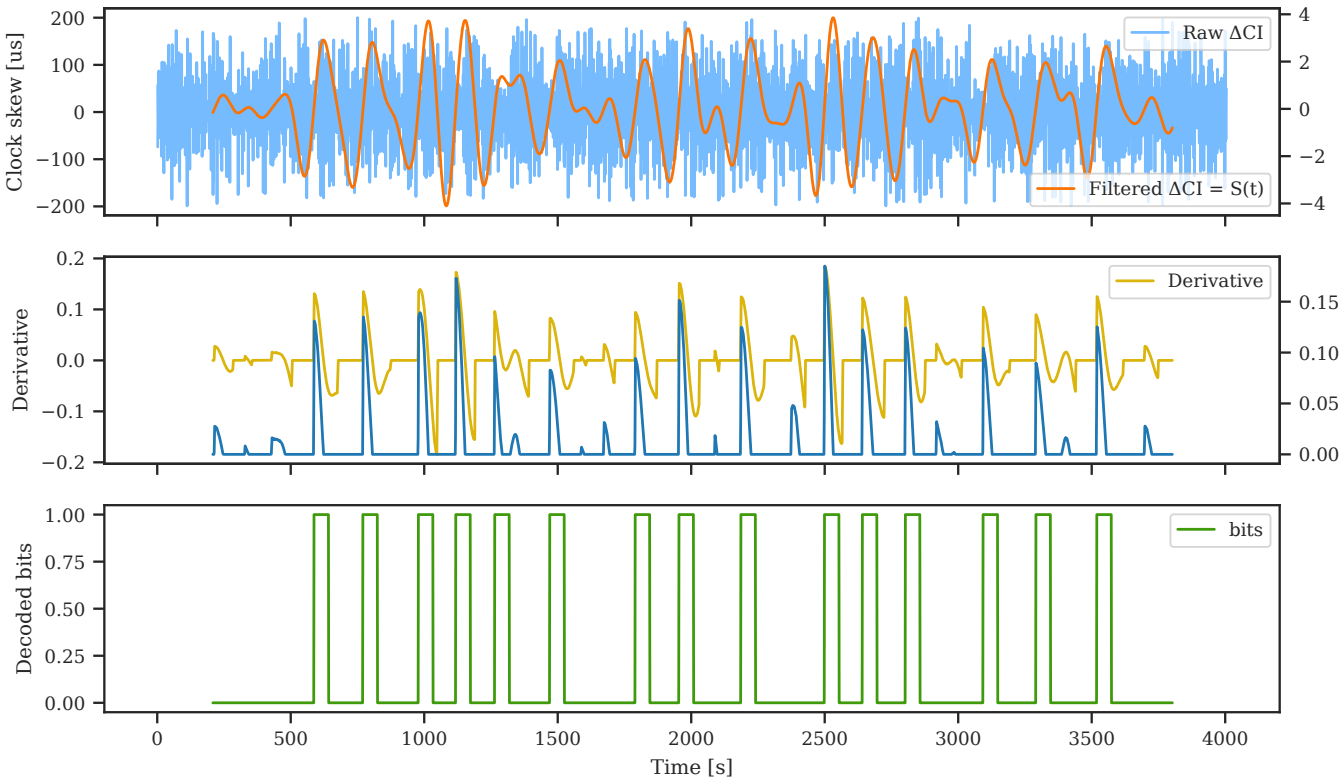

Figure 9: Covert channel on the iPhone 5s. No temperature information was easily available, so the heat graph is omitted.

oscillator. Another important factor is the sensitivity of the crystal to sudden temperature changes. On high-end hardware, such as smartphones, the clock skew variations are suppressed due to the use of a TCXO which hurts the overall throughput of the channel. It makes decoding of the signal more difficult and error prone. Table 4 summarizes the performance obtained on the different platforms.

Table 4: Error rate on different hardware architectures

Platform	Throughput (bph)	Bits sent	Errors	Error rate
RPi 3B	38	61	3	5%
RPi 3B (cooled)	45	33	2	6%
Motorola X	8	6	0	< 1%
iPhone 5s	32	27	0	< 1%

The calibration phase prior to the attack, helps to improve the throughput and accuracy of the channel. Our experiments showed that the exact calibration values are not always available, e.g. on the iPhone no base or target temperature could be detected. In this scenario we experimentally derived and fixed the values for the heat up time and cool down time. By adding an additional margin to these values we can try to compensate situations where the base temperature of the device differs from the situation under which the experimental values where established.

To improve throughput even further we can try to speed up the heating up or cooling down phase. We showed that with the addition of a cooling fan to the control of the malicious application, we can improve the bitrate. Additionally, throughput could be improved by using multi-level encoding techniques if the CPU heat emissions can control the clock skew with acceptable precision and the clock skew variations are sufficiently large.

Compared to the work in [17] our thermal covert channel is slower. Matsi et. al. directly use the CPU temperature to establish the covert channel, which makes it more reactive to changes in CPU load. Our thermal covert channel must indirectly be measured through the effects on the clock skew. The drawback of the work in [17] is the range of the channel. Covert communication is limited to applications running on the same physical machine.

The work presented in Bitwhisper [12] is capable of extending the range of the thermal covert channel to 40 cm. Because of the air-gapped constraints they cannot sniff wireless transmission and therefore the range stays limited.

Our approach has a similar throughput as Bitwhisper but the range of the covert channel extends to the full transmission range of BLE. There is no need for a direct line-of-sight which improves on the attack's stealthiness. Moreover the covert receiver doesn't need to interact with the victim device. It suffices to merely passively sniff the legitimate traffic.

7 CONCLUSION

In this paper we investigated the design of a thermal covert channel in BLE-connections. We leverage the sensitivity of the crystal oscillator to temperature changes. By using well-timed CPU intensive calculations we can encode information under form of a clock skew. A covert receiver sniffs a legitimate BLE connection to measure the BLE connection interval changes. We are guaranteed to have periodic traffic in the legitimate connection due to the need to keep synchronization between the master and the slave.

We implemented our attack on three different hardware platforms. We obtain similar throughput compared to previous work on thermal covert channels, but we succeed in increasing the range of the channel.

In the future we would like to investigate more closely the impact of having multiple CPU intensive applications running at the same moment while the malicious application is encoding data. Additionally we would like to explore the possibility of generalizing the attack approach to fit other low-power IoT protocols that have the same design philosophy as BLE, e.g. IEEE802.15.4e.

ACKNOWLEDGMENTS

This work has been partially supported by the LabEx PERSYVAL-Lab (ANR-11-LABX-0025-01) funded by the French program Investissement d'Avenir, and the FUI IoTize project funded by Région Auvergne-Rhône-Alpes.

REFERENCES

[1] 2012. *IEEE 802.15.4e Low-Rate Wireless Personal Area Networks (Amendment to IEEE Std 802.15.4-2011).* IEEE Standards Office, New York, NY, USA.
[2] Bluetooth SIG 2010. *Bluetooth Core specification 4.0.* Bluetooth SIG.
[3] Bluetooth SIG 2016. *Bluetooth Core specification 5.0.* Bluetooth SIG.
[4] Raj Chandra Bose and Dwijendra K Ray-Chaudhuri. 1960. On A Class of Error Correcting Binary Group Codes. *Information and control* 3, 1 (1960), 68–79.
[5] Cypress Semiconductor Corporation 2016. *BCM4334: Single Chip IEEE 802.11 a/b/g/n MAC/Baseband/Radio with Integrated Bluetooth 4.0 + HS and FM Receiver.* Cypress Semiconductor Corporation. http://www.rumjd.com/Attachments/20160820160827_152216_171.pdf
[6] Cypress Semiconductor Corporation 2018. *CYW43438: Single-Chip IEEE 802.11 b/g/n MAC/Baseband/Radio with Integrated Bluetooth 4.2.* Cypress Semiconductor Corporation.
[7] Atis Elsts, Simon Duquennoy, Xenofon Fafoutis, George Oikonomou, Robert Piechocki, and Ian Craddock. 2016. Microsecond-Accuracy Time Synchronization Using the IEEE 802.15.4 TSCH Protocol. In *IEEE SenseApp 2016-Eleventh IEEE International Workshop on Practical Issues in Building Sensor Network Applications.*
[8] Fluke 2003. *175, 177, 179 True-rms Multimeters: User manual.* Fluke. https://www.instrumart.com/assets/179-manual.pdf
[9] Raspberry Pi Foundation. 2016. Raspberry Pi 3B. https://www.raspberrypi.org/products/raspberry-pi-3-model-b/
[10] Great Scott Gadgets. 2018. Project Ubertooth. https://github.com/greatscottgadgets/ubertooth/
[11] Carles Gomez, Joaquim Oller, and Josep Paradells. 2012. Overview and Evaluation of Bluetooth Low Energy: An Emerging Low-Power Wireless Technology. *Sensors* 12, 9 (2012), 11734–11753.
[12] Mordechai Guri, Matan Monitz, Yisroel Mirski, and Yuval Elovici. 2015. BitWhisper: Covert Signaling Channel between Air-Gapped Computers using Thermal Manipulations. In *28th IEEE Computer Security Foundations Symposium (CSF).* 276–289.
[13] Gareth Halfacree. [n. d.]. Benchmarking the Raspberry Pi 3 B+. https://medium.com/@ghalfacree/benchmarking-the-raspberry-pi-3-b-plus-44122cf3d806
[14] IFIXIT. [n. d.]. iPhone 5s Teardown. https://nl.ifixit.com/Teardown/iPhone+5s+Teardown/17383
[15] IFIXIT. [n. d.]. Motorola Moto X Teardown. https://nl.ifixit.com/Teardown/Motorola+Moto+X+Teardown/16867
[16] APPLE IFIXIT. 2013. The Teardown: Apple iPhone 5s. 8 (November 2013). Issue 10.
[17] Ramya Jayaram Masti, Devendra Rai, Aanjhan Ranganathan, Christian Müller, Lothar Thiele, and Srdjan Capkun. 2015. Thermal Covert Channels on Multi-core Platforms. In *USENIX Security Symposium.* 865–880.
[18] Elodie Morin, Mickael Maman, Roberto Guizzetti, and Andrzej Duda. 2017. Comparison of the Device Lifetime in Wireless Networks for the Internet of Things. *IEEE Access* 5 (2017), 7097–7114.
[19] Steven J. Murdoch. 2006. Hot or Not: Revealing Hidden Services by their Clock Skew. In *13th ACM conference on Computer and communications security.* ACM, 27–36.
[20] Lily Hay Newman. 2018. Inside the Unnerving Supply Chain Attack that Corrupted CCleaner. https://www.wired.com/story/inside-the-unnerving-supply-chain-attack-that-corrupted-ccleaner/
[21] NXP Semiconductors 2015. *LPC1759/58/56/54/52/51: 32-bit ARM Cortex-M3 MCU; up to 512 kB flash and 64 kB SRAM with Ethernet, USB 2.0 Host/Device/OTG, CAN.* NXP Semiconductors. https://www.nxp.com/docs/en/data-sheet/LPC1759_58_56_54_52_51.pdf
[22] Qualcomm Technologies, Inc 2014. *Qualcomm Snapdragon 801 Processor.* Qualcomm Technologies, Inc. https://www.qualcomm.com/media/documents/files/snapdragon-801-processor-product-brief.pdf
[23] Qualcomm Technologies, Inc 2017. *WCN3680B/WCN3660B: device specification.* Qualcomm Technologies, Inc. https://developer.qualcomm.com/download/sd410/wcn3680b-wcn3660b-device-spec.pdf
[24] Magic Blue UU. [n. d.]. Bluetooth Bulb. https://www.gearbest.com/smart-light-bulb/pp_230349.html
[25] Royyan Wijaya, Viktor Vorobyev, and Sandra. 2017. The Noun Project - Icons for Everything. https://thenounproject.com/