USB Snooping Made Easy: Crosstalk Leakage Attacks on USB Hubs

Yang Su, *Auto-ID Lab, The School of Computer Science, The University of Adelaide;*
Daniel Genkin, *University of Pennsylvania and University of Maryland;* Damith Ranasinghe,
Auto-ID Lab, The School of Computer Science, The University of Adelaide;
Yuval Yarom, *The University of Adelaide and Data61, CSIRO*

https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/su

This paper is included in the Proceedings of the
26th USENIX Security Symposium

August 16–18, 2017 • Vancouver, BC, Canada

ISBN 978-1-931971-40-9

Open access to the Proceedings of the
26th USENIX Security Symposium
is sponsored by USENIX

USB Snooping Made Easy: Crosstalk Leakage Attacks on USB Hubs

Yang Su
University of Adelaide
yang.su01@adelaide.edu.au

Daniel Genkin
University of Pennsylvania and
University of Maryland
danielg3@cis.upenn.edu

Damith Ranasinghe
University of Adelaide
damith.ranasinghe@adelaide.edu.au

Yuval Yarom
University of Adelaide and Data61
yval@cs.adelaide.edu.au

Abstract

The Universal Serial Bus (USB) is the most prominent interface for connecting peripheral devices to computers. USB-connected input devices, such as keyboards, card-swipers and fingerprint readers, often send sensitive information to the computer. As such information is only sent along the communication path from the device to the computer, it was hitherto thought to be protected from potentially compromised devices outside this path.

We have tested over 50 different computers and external hubs and found that over 90% of them suffer from a crosstalk leakage effect that allows malicious peripheral devices located off the communication path to capture and observe sensitive USB traffic. We also show that in many cases this crosstalk leakage can be observed on the USB power lines, thus defeating a common USB isolation countermeasure of using a charge-only USB cable which physically disconnects the USB data lines.

Demonstrating the attack's low costs and ease of concealment, we modify a novelty USB lamp to implement an off-path attack which captures and exfiltrates USB traffic when connected to a vulnerable internal or a external USB hub.

1 Introduction

Modern computer systems typically consist of hundreds of components, each with a clear functionality and well-defined input-output interfaces. Connecting all of these components are buses, which transfer information between components. Since the internal hardware components of a system are usually assumed to be trusted, most buses carry no protections against malicious behavior. However, with the development of complicated computer peripherals, buses are no longer kept internal. High-Definition Multimedia Interface (HDMI) [28], DisplayPort [47], the external Serial AT Attachment (eSATA) [43], the Universal Serial Bus (USB) [14], and many others all connect to external devices of unknown origin. Moreover, these buses often carry sensitive information such as key strokes (including passwords), mouse movements, file transfers, screen images, etc.

The security model of these buses does not follow the standard methods of securing communication channels. Rather than using common techniques, such as encryption and authentication, these buses seem to rely on a unicast network model, where messages are physically routed along the path from the sender to the receiver instead of being broadcasted to all of the components connected to the bus. This, coupled with short and simple routes that only have few intermediate components, seems to provide a "good enough" security. As a result, in order to externally monitor traffic such as the victim's keystrokes, the attacker has to corrupt one of the often small number of components that are located between the sender (the keyboard) and the receiver (the USB host). Consequently, it is commonly assumed that "devices are not able to snoop information sent from the Device to Host since information flows only through Hubs until it reaches the Host" [36].

In this paper we challenge this assumption. More specifically, we investigate the following questions:

*Are common communication buses vulnerable to **off-path** attacks? How can such attacks be mounted and at what cost?*

1.1 Our Results

As a case study, in this paper we focus on the Universal Serial Bus (USB) interface, which is the predominant interface used by modern computer paraphernalia. Compared with legacy interfaces such as serial port (RS-232) [17], parallel port (IEEE 1284) [19] or keyboard jack (DIN 41524/IEC 60130-9) [29], USB has wide range of advantages: it is hot pluggable, extensible (via USB hubs) and capable of supporting many types of equipment. A final feature of the USB interface is the ability to provide both data communication and power to peripheral devices.

In this paper we demonstrate that in many cases, data-dependent voltage fluctuations of the USB port's data lines can be monitored from adjacent ports on the USB hub. Our results apply to both internal USB hubs which are installed inside computers, as well as to external off-the-shelf USB hubs. Moreover, this phenomena is not limited to a small number of vulnerable hubs but seems to be quite common, spanning various manufacturers and hub designs. In our experiments, 94% of the internal hubs in computers and in docking stations and 90% of the external USB hubs we evaluated displayed some form of exploitable leakage.

In the context of communication channels, this phenomena is often referred to as channel-to-channel crosstalk [38]. We demonstrate that this crosstalk effect allows an *off-path* attacker to eavesdrop on USB communication. In particular we show that a corrupted peripheral device can monitor the communication of other peripheral devices connected to the same *non-corrupted* USB hub, or just connected directly to the same computer. Moreover, we show that common "ad-hoc" physical protections, such as physically disconnecting USB data and power lines, are often ineffective in stopping the discovered leakage.

Attack Scenario. As noted above, in addition to communication, the USB bus can also provide power to various peripherals. Many "USB toys", such as lamps, fans or office foam rocket launchers [21], often of unknown origin, have been designed to use the feature and are thus routinely connected to USB ports. An attacker can thus augment such a toy with the required equipment in order to monitor the USB port crosstalk and subsequently sell it at below-market-value prices. In Section 6 we show how to cheaply construct such a probe which can monitor and extract the communication of other devices.

We mainly focus on slow-speed USB 1.x input devices, such as keyboards, card readers, fingerprint readers, USB headsets, etc. Information sent from these devices is often sensitive (e.g., passwords, credit card numbers, biometric data, voice conversations, etc.) and thus should remain secret. While faster versions of the USB standard were published almost two decades ago and are in common use, these versions are backwards compatible with USB 1.x and many input devices are manufactured to the slower standard. We believe that in the forseeable future, slow speed-devices will continue to use the USB 1.x interface.

While our proof-of-concept probe (Section 6) was designed to attack USB 1.x devices (connected to any USB hub, including 3.0 hubs), we do show that attacks on devices using the faster USB 2.0 standard are feasible (Section 3.3). We leave the task of attacking USB 3.0 devices connected to 3.0 hubs as an open problem (See Section 7).

1.2 Related Work

For a summary of attacks on USB, see [16, 44] and references therein.

USB Traffic Monitoring. Because USB traffic is not encrypted, on-path devices can listen in to all of the communication that passes through them. This capability is exploited by commercial keyloggers, such as Key Grabber [2] and KeyGhost [1]. Neugschwandtner et al. [34] note that downstream traffic is broadcasted to all devices connected to the bus, demonstrating recovery of all the downstream traffic using a USB analyzer. They further suggest encrypting downstream USB traffic to protect against snooping attacks. Unlike their attack, we capture upstream USB traffic, which is not broadcasted. Furthermore, because their countermeasure only encrypts downstream traffic, it does not prevent our attack.

Oberg et al. [36] describe a timing-based covert channel that creates an off-path information flow between colluding devices. To mitigate the channel they suggest using deterministic time slots for serving each device. We note that our attack allows capture of the actual data transferred from a non-cooperating device and that the suggested mitigation does not protect against our attack.

Exploiting Trust on Buses. Instead of monitoring traffic, malicious devices can attack the host, exploiting weaknesses in the host software [10, 50], firmware [39], trust [15, 5] or protocol [42]. Similarly, malicious hosts can attack attached devices [33, 35, 32, 52]. The attack of the USB bus was also explored by Bratus et al. [12] both at the hardware level and at the device driver level.

To protect the host from malicious devices, Tian et al. [46, 45] and Angel et al. [7] suggest filtering the USB traffic and implementing a permission mechanisms for USB ports. Angel et al. [7] also suggests applying end-to-end encryption between devices and the host to protect the confidentiality and the integrity of USB data in transit.

A common method for protecting hosts from malicious devices and vice versa is to cut the data lines between the two, connecting only the USB power lines. Such an approach allows the host to power a device without the risk of data interchange between the two. Available options for this approach include power-only USB cables as well as dedicated devices such as the USB Condom [3]. We note that such defenses do not protect against our attack in the case that crosstalk leakage is present on the power lines.

Attacks On The Physical Medium. USB Killer [18] is a device designed to collect energy from the USB power line and inject a high voltage pulse back into the computer, to destroy sensitive electronic components.

Vuagnoux and Pasini [48] as well as Wang and Yu [49] show that the electromagnetic (EM) emanations from

PS/2 keyboards can be used to spy on key presses. However, the design of the USB port seems to make this leakage much harder to exploit with only partial information being leaked about key presses, allowing the attack to only narrow down the pressed key to a group of 5 potential keys [48]. EM attacks have also been shown effective in recovering video signals [31] and Ethernet communication [41]. See [22] for a survey of EM-based surveillance attacks. Similar attacks exploit acoustic emanations from keyboards [8, 26] and printers [9].

Side Channel Attacks. Attacks on cryptographic implementations by monitoring devices' electromagnetic emanations and power usage have been extensively demonstrated. See [6, 30] and references therein. While many such works have focused on small devices such as smart cards or FPGAs, recent works have demonstrated similar vulnerabilities in PCs [23] and smartphones [24].

Side Channel Attacks Using USB Ports. The USB ports of various devices were also used for mounting side channel attacks. For laptop and desktop computers, monitoring the USB power lines [37] can reveal information about the system's activity. In addition, the "far-end-of-cable" key extraction attack of [25, 23] can be also mounted over USB ports. For mobile phones, the USB port can be used for power analysis key extraction attacks [24] and distinguishing websites [51].

1.3 Structure of this Paper

The rest of this paper is organized as follows: Section 2 introduces USB, including the bus topology and relevant aspects of its physical and logical protocols. In Section 3 we discuss the crosstalk leakage on the USB data and power lines. We show how to decode the leakage to recover the transferred data in Section 4, and proceed to describe the attacks on various devices in Section 5. Section 6 demonstrates a practical attack using a subverted USB lamp that captures key presses and exfiltrates the information wirelessly via Bluetooth.

2 The USB Interface

USB Versions and Speeds. Since its introduction in 1996, the USB standard underwent three main upgrades. Initially USB 1.x [13] used a single data path supporting up to 127 peripheral devices and with 12 Mbps data rate (also known as USB full-speed). This version currently still powers a huge number HIDs (Human Interface Devices) such as keyboards, remotes and various card readers. Next, in the early 2000's USB 2.0 unified the computer peripheral market, supporting speeds of up to 480 Mbps (also known as USB high-speed) while maintaining backwards compatibility. USB 2.0 is commonly used for devices requiring high data transfer rates, such as external storage devices and Web cameras. Finally, in 2008 another major upgrade of the USB family, USB 3.0, was published [27]. In this version, the maximum bus speed was increased to 5 Gbps (also known as USB superspeed). In order to achieve such a speed and to support full-duplex communication, five new pins were added to the classic connectors and the cable material standard was upgraded.

USB Hubs. USB hubs are commonly used to split a single USB port to many (typically four) ports, thus allowing the user to connect additional peripheral devices. In addition to increasing the number of available USB ports, USB hubs serve four functions. Each USB hub may function as a signal repeater, extending the cable length by five meters. Some hubs may include independent power supply to ensure each downstream port has enough power available. Hubs also function as protocol translators: for example in case a USB 1.0 ticket printer is plugged in a USB 3.0 hub, the hub translates the latest USB 3.0 downstream signal back to the legacy USB 1.0 language and forwards to the printer. Finally, the USB hub also protects the bus by isolating and disconnecting malfunctioning devices which draw too much power or do not obey the USB protocol.

USB Tiered Topology. All USB devices are connected in a tree topology, up to 127 devices (including any hubs) can be connected. At the root of the tree, there is a single host (also known as USB root hub) which is directly addressable from CPU. The host coordinates the USB tree network and in USB 1.0 and 2.0 it is the only one in the network who can initiate communication. Up to five additional hubs can be cascaded in series on each tree branch. Each hub has one upstream port and up to seven downstream ports. Downstream traffic is broadcasted to all of the devices in the tree. However, upstream data is only sent along the (single) path from the transmitting peripheral device to the host. In particular, hubs which are not located on the path between the transmitting peripheral and the host should not be able to observe the peripheral's upstream USB traffic.

Broadcasting USB downstream traffic is a risky design decision [34]. For example, an attacker can use a simple USB analyzer to monitor disk writes. However, because upstream traffic is only transmitted along the path to the host, much of the "interesting" data, such as keyboard inputs and disk reads, seems to remain inaccessible to a corrupted peripheral device outside that path.

USB 1.x and 2.0 Ports Structure. Both USB 1.x and USB 2.0 use a two-wire differential communication bus. We denote these wires as D+ and D-. In addition, each USB connection also provides two power lines, denoted as Vcc and GND, to supply 5 Volt power to peripheral devices. See Figure 1.

USB 1.x and 2.0 Communication. Sharing the bus is achieved through the use of Time-Division Multiplexing

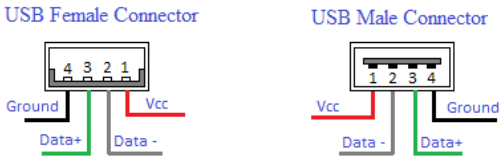


Figure 1: Structure of a USB port.

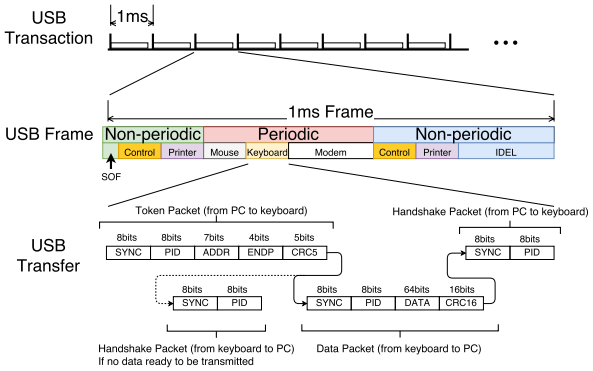


Figure 2: USB transaction, frame and packets.

(TDM). The bus protocol divides time into 1 ms *frames*, as shown in Figure 2. Each frame comprises a Start of Frame (SOF) packet followed by several *transfers*. Transfers can be *periodic*, repeating in every frame as long as the target device remains connected to the bus. These are used, for example, for polling input devices which are supposed to update as frequently as possible.

Otherwise, transfers are *non-periodic*. These include control transfers assigning a bus address to a newly plugged device, or occasional data-transfers such as updating the print queue of a printer. Each transfer consists of packets, which are the smallest building block of USB traffic. At the bottom of Figure 2, we illustrate a keyboard report transfer. The transfer begins with a *Token* packet that probes the keyboard for key presses. If undelivered key presses are present, the keyboard responds with a *Data* packet that contains a 64 bit payload identifying the pressed key. The host then responds to the Data packet, sending a *Handshake* packet, which terminates the transfer. If no key presses are available for delivery, the keyboard responds to the initial Token packet with a Handshake packet that terminates the transfer.

To protect data from corruption, packets include several checksum mechanisms. The Packet Identifier (PID) field is protected by requiring that the second half of the field is the bitwise complement of the first half. Token packets transmitted from the host to the device use a 5 bits CRC (Cyclic Redundancy Check) to verify the Address (ADDR) and Endpoint (ENDP) fields and Data packets use a 16 bits CRC in order to verify the payload.

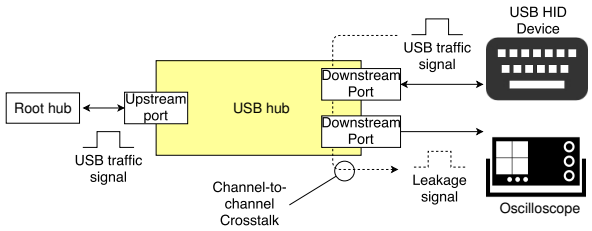


Figure 3: Leakage can be observed at an unused port adjacent to the USB device.

3 Leaky Hubs

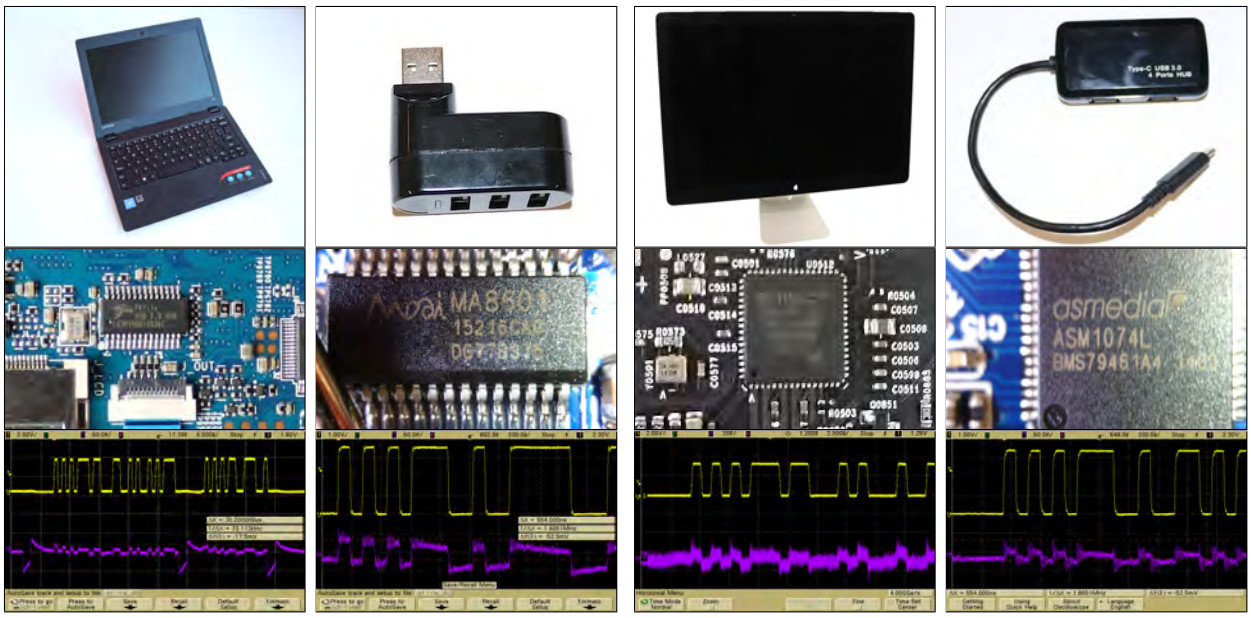
We now turn our attention to the leakage between two adjacent USB ports on the same USB hub. We investigate both internal hubs, installed inside computers, and external stand-alone hubs. As mentioned in Section 2, downstream traffic from the PC to the peripherals connected to the hub is broadcasted and thus readily available. However, upstream traffic, e.g. keyboard presses, is not broadcasted and thus should remain out of reach for an attacker monitoring the USB port. We evaluated the crosstalk leakage between two downstream USB ports on the same USB hub as follows. First, we connected a USB input device, such as a keyboard, to one of the ports of the hub. Next, we used an oscilloscope to monitor the data sent from the device to the host while concurrently measuring the leakage on a *different* USB port of the same USB hub. See Figure 3.

As we described in Section 2, every USB port contains a pair of data lines and a pair of power lines. Each of these pairs is a potential source of leakage. Hence, we can measure the crosstalk leakage present on the data lines of an adjacent port and additionally, or alternatively, we can measure it on power lines of the adjacent port.

In a typical scenario both data line and power line leakage should be available on the same USB port, thus allowing the attacker to choose the channel containing the best signal. However, common "ad-hoc" countermeasures against untrusted USB devices are sometimes deployed. These include USB hubs with dedicated switches, which power devices down by cutting the power supply to them and power-only cables, a.k.a. *USB condoms* [3], which disconnect the data lines in order to prevent interaction between the device and the USB host. Yet, because the crosstalk leakage is often present on both the power and the data lines, to completely render the attacker ineffective, *both* pairs should be disconnected.

3.1 Data Line Leakage

Experimental Setup. In order to evaluate the data line crosstalk leakage present on USB hubs we used an Agilent MSO6104A oscilloscope (1GHz, 4Gsps) and two Agilent 10073C 500MHz passive probes. We then con-



(a) Terminus Tech FE1.1s (inside a Lenovo 100s laptop)

(b) Prolific MA8601

(c) SMSC USB2517-JZX (inside an Apple Display)

(d) ASMedia Technology ASM1074L

Figure 4: Top and middle row: Four tested USB hubs and their controller chips found to contain data line crosstalk leakage. Bottom row: Corresponding leakage waveform, yellow (top) trace shows the USB traffic and purple (bottom) trace shows the data line crosstalk leakage measured from an adjacent downstream USB port.

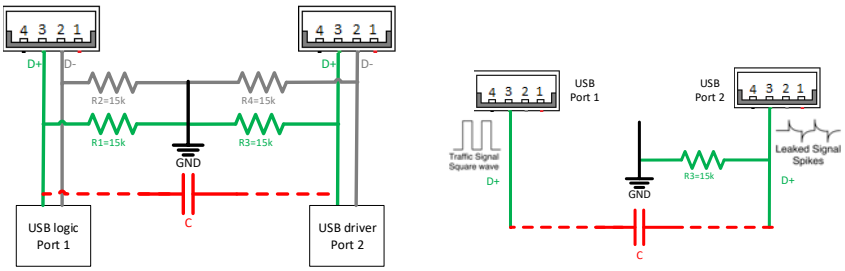


Figure 5: Typical schematic of a USB hub (left) and the RC differentiator created by C and $R3$ (right).

nected a USB keyboard (Lenovo KU-0225) to one of the hub's downstream ports. Next, we used one of the oscilloscope's probes to monitor the communication between the PC host and the keyboard by measuring the voltage on the D+ line relative to the GND line. Finally, we observed the data line crosstalk leakage (using the oscilloscope's second probe) by measuring the voltage on the D+ or D- line relative to the GND line on one of the hub's other downstream ports.[1]

Observing the Data Line Crosstalk. Figure 4 shows four different devices, including both leaking computers and leaking external hubs. The correlation between the actual keyboard data (yellow trace, top) and observed data line crosstalk leakage (purple trace, bottom) can be clearly seen. We find that such data line crosstalks are quite common. We evaluated 34 internal and 20 external USB hubs. Out of these, 17 internal and 17 external were found to have a data line crosstalk. Finally, we note that data line crosstalk is not limited to USB 2.0 ports and is also noticeable on USB 3.0 ports (see Figure 4(d)).

Leakage Mechanism. The leakage waveform of Figure 4(a) provides a hint into the physical reason for the existence of crosstalk between two different USB ports. A typical hub controller chip contains four USB logic blocks, each responsible for a single downstream USB port. See Figure 5. As part of the speed negotiation between the hub and downstream devices, the data lines of the USB port are pulled down using $15k\Omega$ resistors. (These are marked as $R1, R2, R3, R4$ in Figure 5.) Theoretically, the data lines of USB port 1 should be completely isolated from those of USB port 2. However, we conjecture that the close proximity of the USB logic blocks inside the controller chip creates some parasitic

[1] The choice between measuring the D+ or D- relative to the GND line seems to depend on the specific port and hub used. In each experiment below we actually attempted both options and present the option which showed the clearest signal.

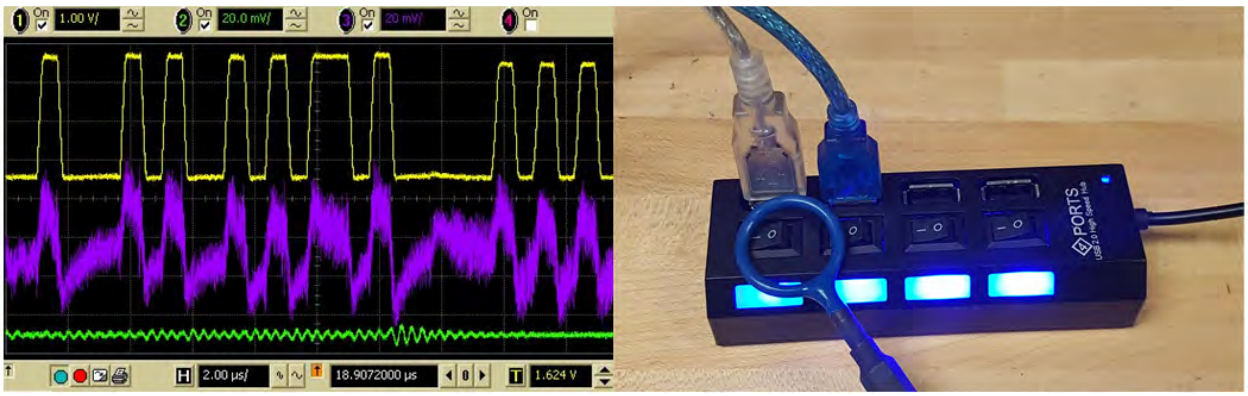


Figure 6: Monitoring the hub's electromagnetic leakage as well as the data line crosstalk leakage. The right figure is the experimental setup where a keyboard is connected to the hub's leftmost port (silver wire), the crosstalk leakage is monitored via the adjacent port (blue wire) and the electromagnetic field is measured using an EM probe (blue loop). The left figure shows the corresponding signals where the top (yellow) trace is the actual USB data, the middle (purple) trace is the data line crosstalk leakage and the bottom (green) trace is the observed EM signal. Although the crosstalk signal is plotted with the same vertical scale as the EM signal, only the crosstalk signal (middle, purple) exhibits a clear correlation with the actual USB data (top, yellow).

capacitance between data lines of adjacent ports (see C in Figure 5). Thus, any signal present on D+ line of port 1 passes through the RC differentiator created by C and $R3$ and can be observed on the D+ line of port 2. See Figure 5. Similar crosstalk leakage also happens with the D- lines with one option typically giving much better signal than the other, depending on the USB hub and on individual ports in it.

Crosstalk or EM? In order to ascertain that the observed leakage indeed emanates from crosstalk and not from electromagnetic interference, we have used an EM probe (Langer LF R400) to measure the electromagnetic field emitted by an external USB hub. As can be seen in Figure 6, while there is a clear correlation between data line crosstalk leakage (purple, middle) and the real USB traffic (yellow, top), the hub's electromagnetic radiation (green, bottom) does not contain any observable information. We thus conclude that the observed crosstalk leakage indeed emanates from parasitic capacitance between the hub's USB ports and not from the hub's electromagnetic leakage.

3.2 Power Line Crosstalk Leakage

A common method for isolating potentially corrupted USB devices while still supplying them with 5V power is to physically disconnect the USB data lines. Indeed, power-only USB cables and USB condoms guarantee to isolate corrupted devices from the USB bus while still allowing the use of the USB port as a source of power, e.g. for plugging a mobile phone into an untrusted charging station. In this section we show that by monitoring the *power lines* of a USB port, it is possible to eavesdrop on the communication of USB devices connected to different USB ports. Thus, even if the attacker is connected to the hub using a power-only USB cable, he can still observe the communication of nearby USB devices.

Experimental Setup. We connected a USB keyboard to one of the hub's downstream ports. We then used one of the oscilloscope's probes in order to monitor the communication between the PC host and the keyboard by measuring the voltage on the D+ line relative to the GND line. Finally, using the oscilloscope's second probe, we observed the power line crosstalk leakage by measuring the voltage on the Vcc line relative to the GND line on one of the hub's other downstream ports.

Observing the Power Line Crosstalk Leakage. Figure 7 shows four devices along with the observed signals, confirming the existence of power line crosstalk leakage. The correlation between the actual keyboard data (yellow, top) and the observed power line crosstalk leakage (blue, bottom) is clearly visible. Overall, we found that 29 of the 34 internal and 17 of the 20 external hubs we tested show power line crosstalk leakage. Overall, 32 internal hubs and 18 external hubs show at least one type of crosstalk leakage.

Evaluating USB Condoms. We have also examined the crosstalk leakage present on the USB power lines measured through a PortaPow USB condom [4] which promises to "block data transfer to / from a computer, preventing data security breaches and viruses / hacking when charging from a public USB socket". As can be seen in Figure 8, the power line crosstalk leakage can be clearly observed.

Leakage Mechanism. Parasitic capacitances are present not only between two proximate data lines, but also exist across a data line and a nearby USB power line. See Figure 9 with the parasitic capacitance marked as $C1$.

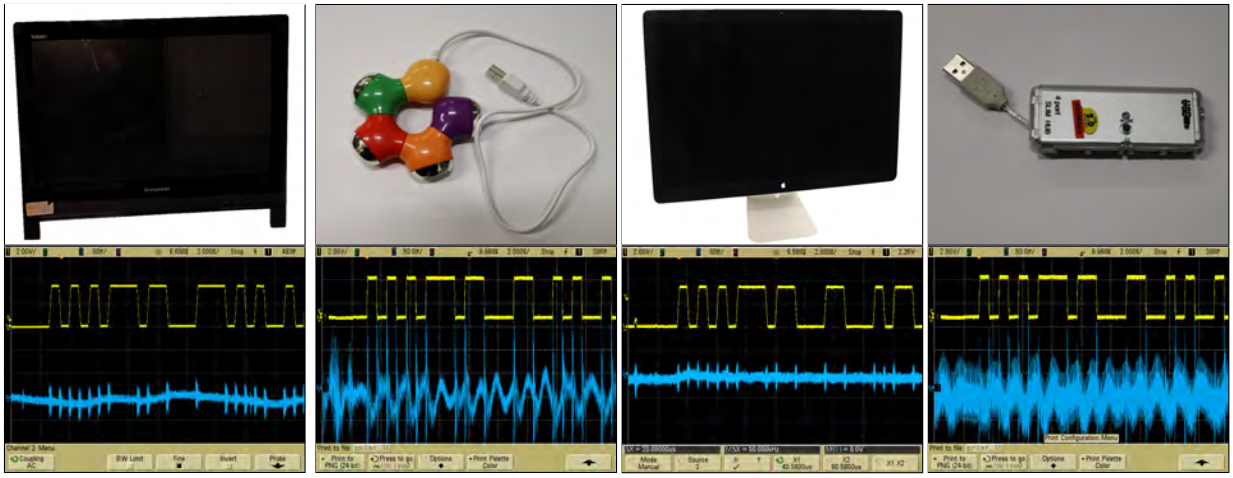


Figure 7: Top row: four tested USB hubs found to contain power line crosstalk leakage. Bottom row: Corresponding leakage waveform, yellow (top) trace shows the USB traffic and blue (bottom) trace shows the power line crosstalk leakage measured from an adjacent downstream USB port. Notice the correlation between the sharp spikes in the leakage trace and the USB traffic. The waveforms were captured using an Agilent MSO6104A oscilloscope (1GHz, 4Gsps) and two Agilent 10073C 500MHz passive probes.

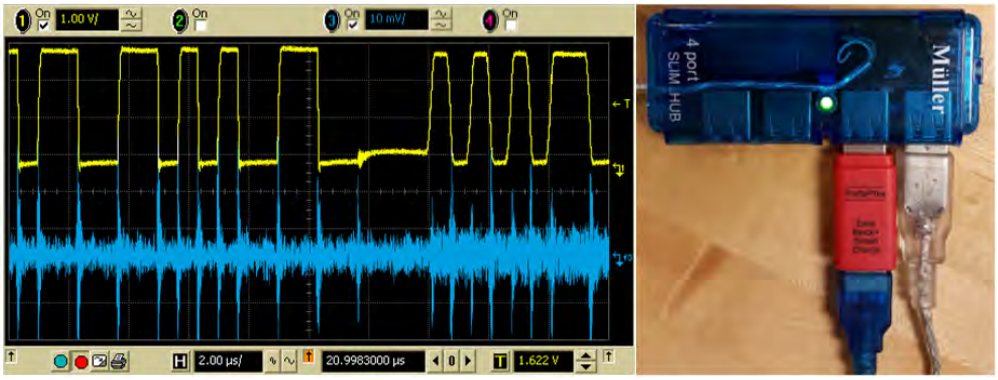


Figure 8: Measuring powerline crosstalk leakage through a PortaPow USB condom. The right figure is the exprimental setup where a keyboard is connected to the USB hub via the silver wire and the powerline crosstalk leakage in monitored through the PortaPow USB condom (red) via the blue wire. The left figure shows the correponding signals (acquired using an Agilent Infiniium DSO 5454832B oscilloscope) where the top (yellow) trace is the actual USB data and the bottom (blue) trace is the powerline crosstalk leakage. Notice the clear correlation between the two traces.

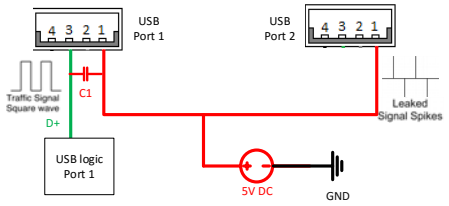


Figure 9: Power line crosstalk leakage mechanism. The parasitic capacitance is marked by $C1$.

Next, since the power lines of all of the USB ports are generally interconnected inside the hub controller chip, the data-to-power crosstalk occurring in one port can be also observed from another port.

3.3 Attacking USB 2.0 Devices

So far we have mainly focused on crosstalk leakage created by USB 1.x devices (such as keyboards or other human interface peripherals). Similar effects are also present with high speed USB devices, such as USB storage and webcam devices. However, since these devices operate at a much greater speed, the experimental setup used in Section 3.1 is no longer sufficient.

Experimental Setup. In order to observe the data line crosstalk leakage from high speed USB 2.0 devices, we used an Agilent DSO 90404A oscilloscope (6GHz, 20Gsps). We then connected a USB drive to one of the hub's downstream ports and used an Agilent N2795A active probe in order to monitor the communication be-

Figure 10: USB 2.0 data line crosstalk leakage from a hub of the same make as the one used in Figure 6. The yellow (top) trace shoes the USB traffic and the blue (bottom) trace shows the observed data line crosstalk leakage measured from an adjacent downstream port (manually aligned with the yellow trace by subtracting 4ns).

tween the USB drive and the PC host while transferring files from the USB drive *to* the PC host. Finally, we used an Agilent N2752A differential probe in order to monitor the voltage between the D+ and D- lines on one of the hub's other downstream ports.

Observing USB 2.0 Data Line Crosstalk Leakage. Figure 10 shows that resulting data line crosstalk leakage while transferring data from the USB drive to the host PC. The correlation between the actual data (yellow, top) and the observed data line crosstalk leakage (blue, bottom) can clearly be seen. We recall that while USB downstream traffic (from the PC host to the USB devices) is broadcasted, USB upstream traffic (such as transferring files from the USB drive to the PC host) is not broadcasted. Thus, it should not be possible to observe the data being transferred from the USB drive to the host PC.

4 Leakage Decoding

4.1 Decoding USB Traffic

Physical Layer. As mentioned in Section 2, both USB 1.x and USB 2.0 use a two-wire differential communication bus, whose wires we denote by D+ and D-. Theoretically, the voltage of D+ relative to D- should be one of two values, either 'high' ($3.3V$ for USB 1.x and 300mV for USB 2.0) or 'low' ($-3.3V$ for USB 1.x and -300mV for USB 2.0).

Non Return to Zero Inverted (NRZI) Encoding. Both USB 1.x and USB 2.0 use NRZI encoding in order to transmit individual bits across the communication bus. One bit is transmitted in each clock cycles, with zeroes represented at the physical layer as transitions between the low and the high voltage levels whereas ones are represented as a lack of transition, i.e. keeping the voltage constant across the clock cycle. To maintain clock synchronization, the USB bus avoids long periods of no transitions using *bit stuffing* encoding [11]. More specifically, it inserts a zero after every sequence of six consecutive ones. At the receiving end, voltage transitions are used to maintain clock synchronization. The receiver otherwise ignores the artificially inserted zeroes.

Decoding USB Packets. Figure 11 presents a USB transfer between a host and a USB keyboard. It shows the signal that represents the communication (blue, top) alongside the corresponding leakage captured on the data lines of another port of the hub (green, bottom). The transfer consists of a clock synchronization followed by a token packet from the host, requesting information about keyboard presses. Following the token, we see the keyboard's response which contains a payload with the key press information.

Field	SYNC	PID	ADDR	ENDP	CRC5
value	00000001	10010110	0101100	1000	10001
Comment	–	IN	0x0A	0x01	–

Note that the two halves of the PID field (1001 0110) complement each other, signifying that the PID check is correct and making it an incoming (IN) packet from the PC to an attached peripheral with address ADDR at endpoint ENDP. Next, as mentioned in Section 2, the token packet also contains a CRC5 field (using the polynomial $X^5 + X^2 + X^0$) of the ADDR and ENDP fields. Indeed, performing long division of 010110 1000 over 100101 gives 10001, which is exactly the CRC5 field of the packet. Finally, at the right bottom corner of Figure 11 there is a clip of payload carried in the DATA packet whose value is 00010000. Since USB data is transmitted least significant bit first, the transmitted value is **0x08**. This scancode matches the "**E**" key on the keyboard indicating that this key was pressed.

4.2 Decoding Data Line Crosstalk Leakage

We now present the signal processing techniques we use to decode the information available via data line crosstalk leakage. As Figure 11 shows, there is a clear correlation between the data line crosstalk leakage (green, bottom) and the actual USB communication (blue, top). However, the transition between signal levels in the leakage trace are less clear than in the communication trace. In order to automatically and reliably decode the information present in the data line crosstalk leakage trace we have performed the steps outlined below. These steps are carefully chosen to allow implementation on cheap and simple hardware that an adversary can conceal easily. See Section 6.

Step 1: Leakage Trace Cleanup. As can be seen in Figure 12(top, black), the data line crosstalk leakage trace contains high frequency noise, making detecting the bus transitions difficult. In order to remove this high frequency noise, we have applied triangular window

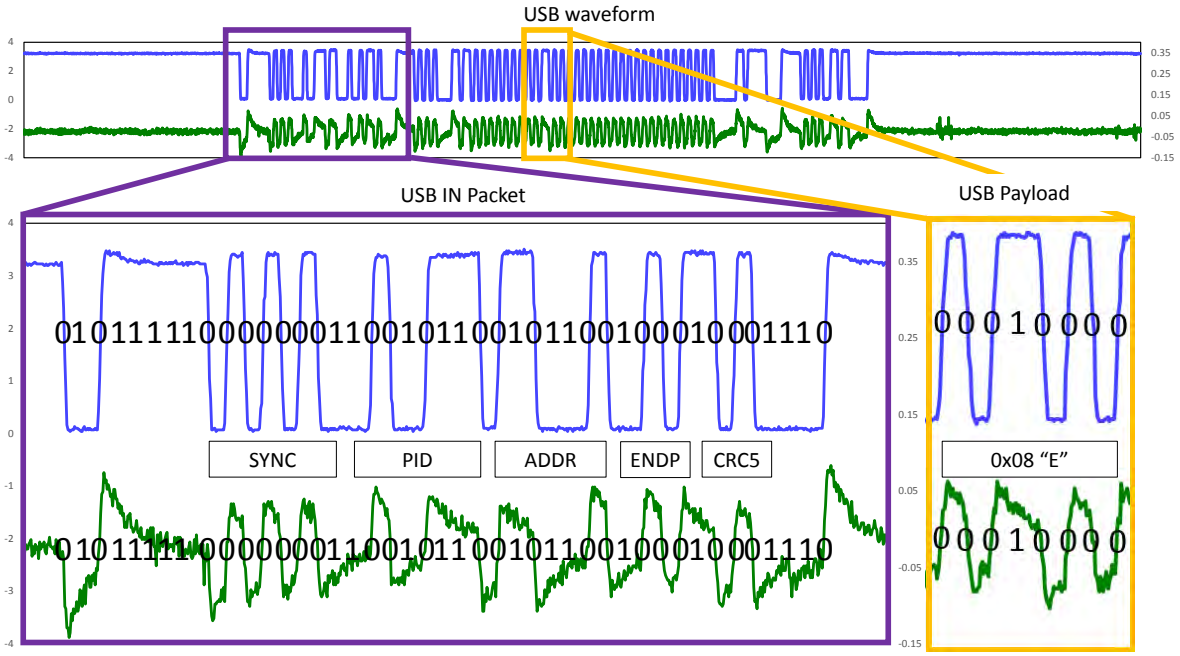


Figure 11: USB communication (blue, top) and data line crosstalk leakage (green, bottom) of a USB frame with a keyboard attached to the USB hub. The data line crosstalk leakage was captured using the hub from Figure 6.

moving average filter. This removed the high frequency spikes in the leakage trace, see Figure 12(top, blue).

Step 2: Transition Enhancement. To precisely locate the bus transitions in the trace we produced at Step 1, we calculated its derivative. That is, we want to find $V' = \frac{\partial V}{\partial t}$ where V denote the trace produced by Step 1 above. Simplifying this operation, we approximated the derivative by subtracting from each sample at some location t in V the sample present at location $t-10$ in V.[2] Figure 12(middle) is the result of this derivative approximation. Note that the rising edges appear as local maxima and the falling edges appears as local minima.

Step 3: Edge Detection. As mentioned in Section 4.1, in the NRZI encoding the rising and the falling edges are equivalent. Both represent a level toggle in physical layer, which corresponds to a transmission of a zero bit. Thus, we first compute the absolute value of the trace we produced in Step 2, see Figure 12(black, bottom). Next, in order to decode the data line crosstalk leakage accurately, we need to know the exact times of the trace's edges. A naive approach would be to attempt to locate all the local maxima of Figure 12(black, bottom). However this approach is unreliable as it might be distracted by any noise, such as the glitches between samples 200 and 250 in Figure 12(black, bottom).

Instead, we apply a simple thresholding to locate the edges. As Figure 12(blue, bottom) shows, we used a fixed threshold of 0.048 V.[3] Next, every time the trace (black) crosses the threshold low to high we consider this to be a transition of the physical layer. See Figure 12(green, bottom).

Step 4: NRZI Decoding. As mentioned in Section 4.1, in the USB protocol uses NRZI encoding. More specifically, the value of a transmitted bit is indicated by maintaining a fixed signal level for a logical one and a transition between signal levels for a logical zero. To decode the signal we use the timing of physical layer transitions to find zeroes (Figure 12(green, bottom)). Next we use the length of the intervals between transitions to find the number of ones. Finally, to account for bit stuffing, we remove any logical zero appearing after six consecutive logical ones.

4.3 Decoding Power Line Crosstalk Leakage

We now turn to the signal processing techniques we use to decode the information available via power line crosstalk leakage. As can be seen from the red trace in Figure 13, every transition between the high and low levels on the USB communication lines creates a short, sharp glitch on the USB power lines. Thus, to decode the information present in the power line crosstalk leakage we performed the following.

As in Step 2 of Section 4.2, we approximated the first

[2] Note that $(V_t - V_{t-n})/n$ is a discrete approximation of the derivative at point t. In our case setting $n = 10$ seems to produce the best results. Note further that because n is constant, discarding the division does not affect the overall shape of the trace.

[3] This value was set empirically and may vary between different leaky hubs.

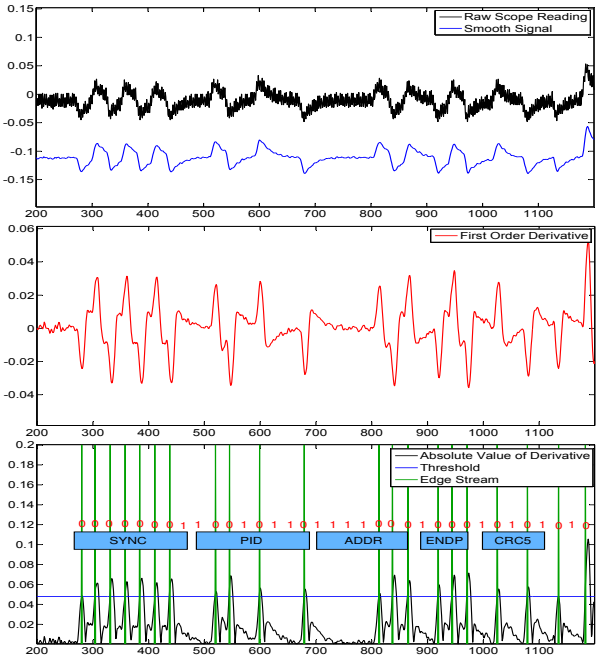


Figure 12: Trace transformations performed to decode the data line crosstalk leakage. (top, black) is the raw data line crosstalk leakage from the hub in Figure 6 and (top, blue) is the result of removing the high frequency noise done in Step 1. (middle, red) is the result of applying the trace enhancement step (Step 2) on the results of leakage trace cleanup step (Step 1). Finally, the (bottom, black) trace is the absolute value of the (middle, red) trace and the green markings denote all locations where the bottom black trace crosses the blue threshold.

order derivative by subtracting each sample from the previous one. We then computed the absolute value of the approximated derivative and smoothed the resulting trace using a moving average filter of 10 samples. This resulted in a relatively clean trace with the toggling in physical layer clearly visible as sharp spikes. See Figure 14(blue). We then applied Steps 3 and 4 from Section 4.2 using a threshold of 0.05 V for edge detection. This resulted in a clear detection of the physical layer toggling events. See Figure 14(green).

5 Leakage Crosstalk Attacks

In this section we present several crosstalk leakage attacks against various peripheral USB devices.

Experimental Setup. We used an Agilent MSO6104A oscilloscope (1GHz, 4Gsps) with Agilent 10073C 500MHz passive probes to monitor the communication between the attacked peripheral and USB host while at the same time monitoring either the data line or power line crosstalk leakage.

Attacking USB Keyboards. Using data line crosstalk leakage, we have successfully extracted keyboard presses from USB keyboards, see Figure 15. Similar results were obtained using power line crosstalk leakage. Finally, notice that in this case, the USB hub had two additional USB devices connected to it (a USB mouse and a USB headset) in additional to the USB keyboard. Nonetheless, we have successfully extracted the keyboard presses, despite additional USB traffic from other devices, functioning concurrently to USB keyboard.

Attacking USB Magnetic Card Readers. In addition to USB keyboards, we have successfully extracted credit-card data from a USB magnetic card reader (MagTek 21040140) using data line crosstalk leakage from an internal USB hub of a Lenovo Ideapad 100s laptop. See Figure 16 for a picture of the experimental setup and Figure 17 for the extracted data. Similar results were also obtained using power line crosstalk leakage.

Attacking USB Headsets and USB Fingerprint Readers. Two other types of devices we successfully attacked are USB headsets and USB Fingerprint Readers. For the headsets, we captured the signals corresponding to the microphone (see Figure 18). We have also successfully observed and decoded the USB communication of a USB fingerprint reader during a finger swipe (see Figure 19). We did not attempt to decode either the voice communication of the headset or the fingerprint data because these devices use propriety data-transfer formats, and reverse engineering these is beyond the scope of this paper. However, we did recover the USB traffic and with the knowledge of the protocols, interpreting the captured data should be straightforward.

Attacking USB Storage. In addition to attacking human interface devices, we have also mounted crosstalk leakage attacks on USB 1.1 drives connected to both internal and external USB 2.0 hubs. Indeed, we have successfully recovered the communication during a file transfer from a USB 1.1 drive to the PC host using data line crosstalk leakage with both external and internal USB 2.0 hubs. See Figure 20. Due to the complexity of the USB driver stack and the file system, we did not attempt to decode the obtained traffic. However, we claim that since the USB communication was completely recovered from the crosstalk leakage, recovering the transferred file can be achieved as well. Finally, similar results were also obtained using power line crosstalk leakage.

6 Exploiting Crosstalk Leakage via Malicious Peripherals

In this section we show how to construct a malicious peripheral device (spy probe) which can successfully extract USB keyboard presses from the data line crosstalk leakage. After extraction, the spy probe exfiltrates the key presses via Bluetooth.

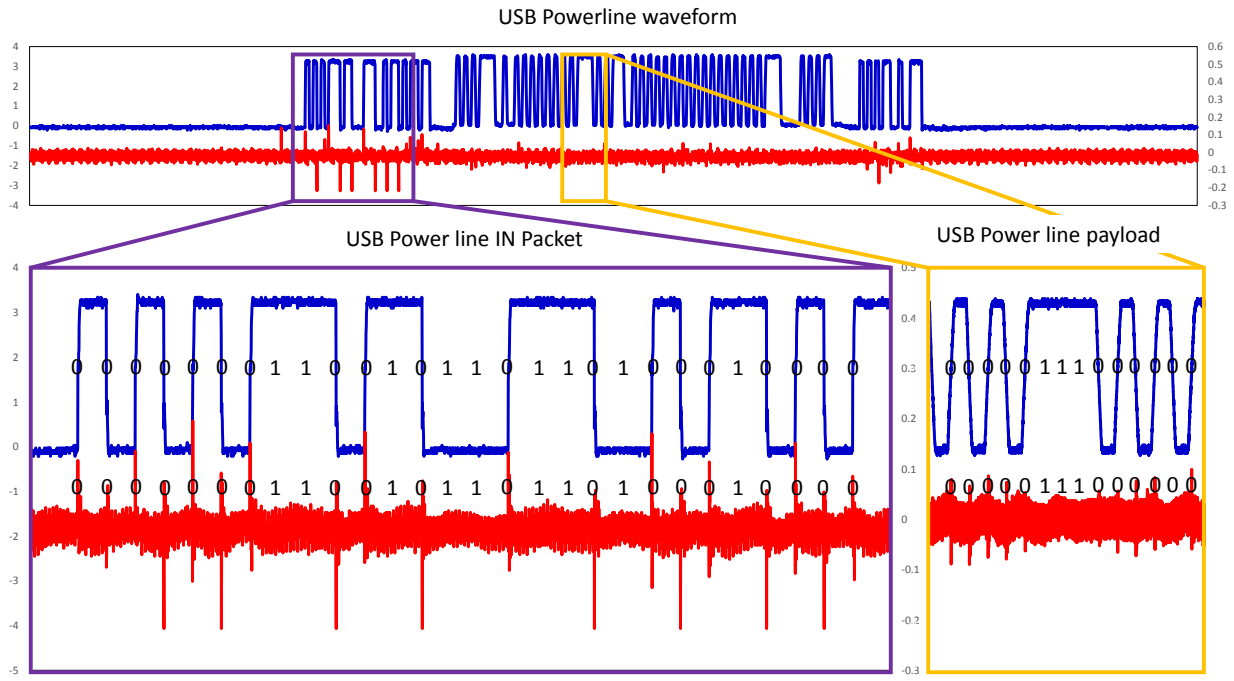


Figure 13: USB communication (blue, top) and power line crosstalk leakage (red, bottom) of a single 1ms USB frame with a keyboard attached to the USB hub. The data line crosstalk leakage was captured using the hub from Figure 7(rightmost).

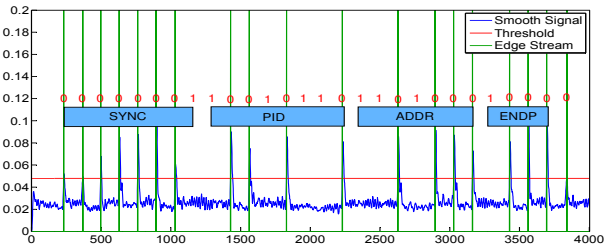


Figure 14: Smoothed trace (blue) subjects to the threshold (red) and the resultant Edge detected (green).

Figure 15: Extracting keyboard presses using data line crosstalk leakage. The scan code 0x07 corresponding to the letter d is clearly visible in the leakage trace.

6.1 Design Overview

Hardware. The spy probe is constructed from an Alinx AX309 FPGA development board (30 USD) connected to an AN108 analog to digital conversion (ADC) board (15 USD) designed by heijin.org. Data is exfiltrated using a WeBee Bluetooth Low Energy (BLE) Board with a Texas Instruments CC2540 chip (5 USD). All of the probe's components are concealed in a USB ghost lamp (20 USD). See Figure 21. In case the 5V USB power is not available (such as in the case where the power lines are disconnected in an attempt to isolate a malicious device), the lamp also contains a battery pack.

The ADC Board. We have connected the ADC board to a male A-type USB plug which should be plugged into the leaky USB hub in order to monitor the data line crosstalk leakage. We have connected the ADC's input to the D+ USB line and monitored its voltage relative to the GND line. We have also used the USB's 5V power line in order to power the probe.[4] Our ADC board has a clamp circuit, attenuator (AD8065), low pass filter and an 8 bit 32 MSaps ADC in a chain. The clamp is a protective element consisting of two germanium diodes, to ensure that the voltage of the signal feed into the ADC never goes above 5 Volts or below GND. Immediately after the clamp there is an attenuator, mapping the input signal of $\pm$ 5 Volts into a 0–2 Volt range. In order to remove high frequency noise, there a simple RC low-pass filter ($f_c = 723MHz$) between the attenuator and the ADC. Finally a AD9280 ADC is used to digitize the data

[4] As mentioned above, the spy probe also contains a battery pack for the case where the 5V power is not available.

Figure 16: (left) Experimental setup for extracting credit card data from a USB card reader (MagTek 21040140 card reader and a Lenovo Ideapad 100s laptop) using data line crosstalk leakage (clearly visible on the oscilloscope's screen) from an internal USB hub. (right) MagTek 21040140 USB magnetic card reader.

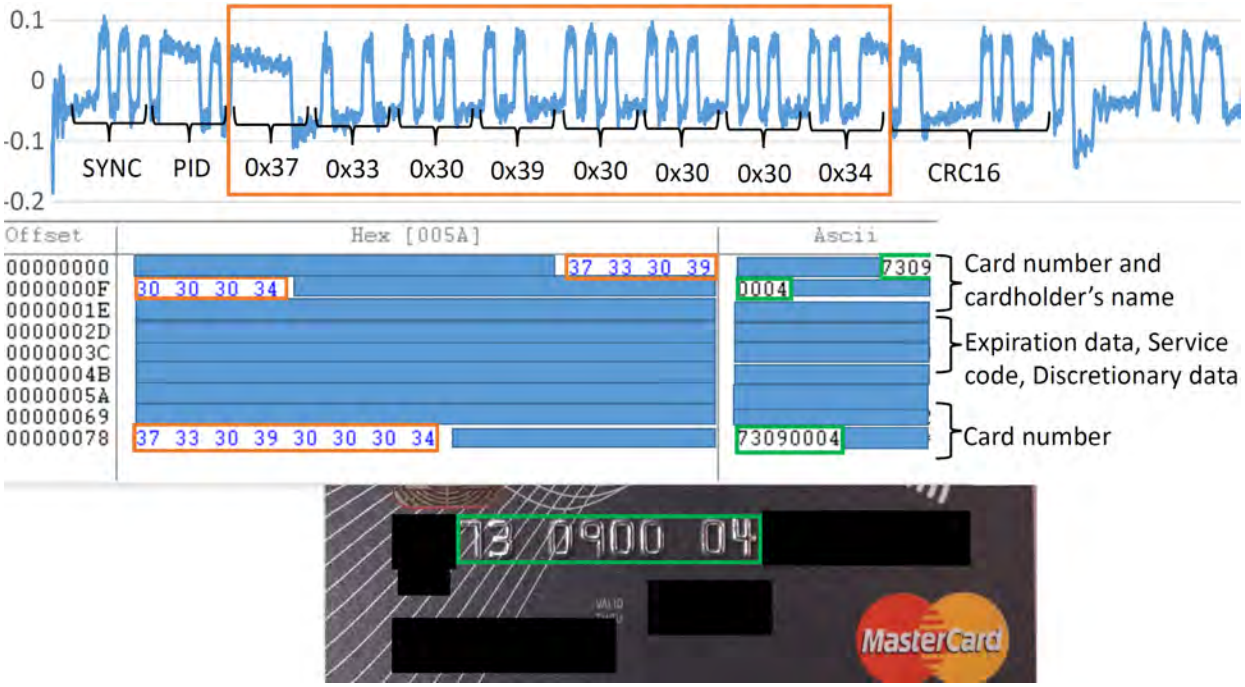


Figure 17: Extracted credit card data using data line crosstalk leakage. (top) Observed data line crosstalk leakage trace segment. Part of the credit card number is visible in hexadecimal encoding (marked in orange box). (middle) hexadecimal to ascii conversion of the extracted data. Part of the credit card number is visible in ascii form (green). (bottom) picture of the credit card used. Notice the correct extraction of the credit card number. In order to protect owner's privacy we have hidden all other card details.

line crosstalk leakage signal. The ADC receives its clock from the FPGA board and transmits 8 bits of data per sample back to the FPGA board. Because the signals we measure are typically $30mV$ peak to peak, we bypassed the attenuator with a jumper cable thereby improving the measurement resolution. See Figure 22.

Software. In order to decode the data line crosstalk leakage recorded by the probe's ADC board, we have implemented a highly optimized version of the signal processing approach described in Section 4.2 on the probe's FPGA board, in Verilog HDL. After decoding the data line crosstalk leakage, the spy probe filters out USB packets which correspond to keyboard presses and exfiltrates them via a bluetooth connection.

6.2 Attack Performance

In this section we evaluate our spy probe's ability to correctly recognize and exfiltrate USB keyboard presses.

Experimental Setup. We used a Microsoft SurfacePro laptop as a USB host. This machine has only one USB slot, forcing the end user to use an external USB hub in order to simultaneously connect a keyboard and mouse.

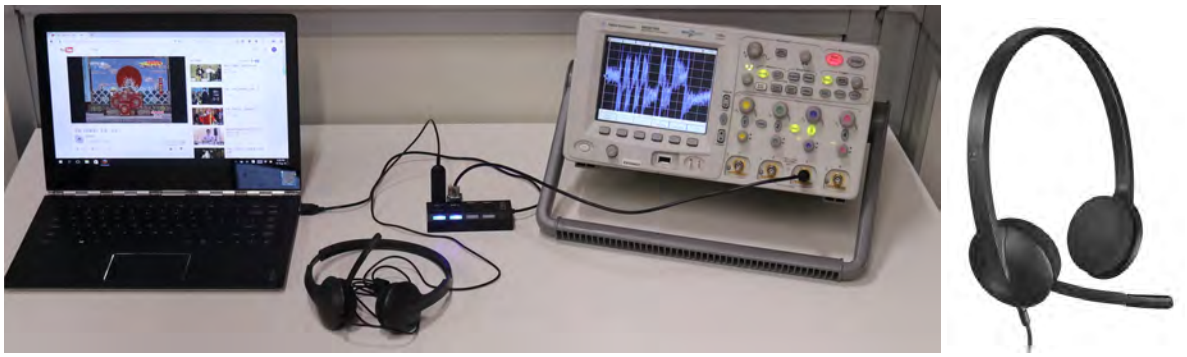


Figure 18: (left) Experimental setup for observing data line crosstalk leakage from a USB headset microphone (Logitech H340). The data line crosstalk leakage is clearly visible on the oscilloscope's screen. (right) Logitech H340 USB headset.

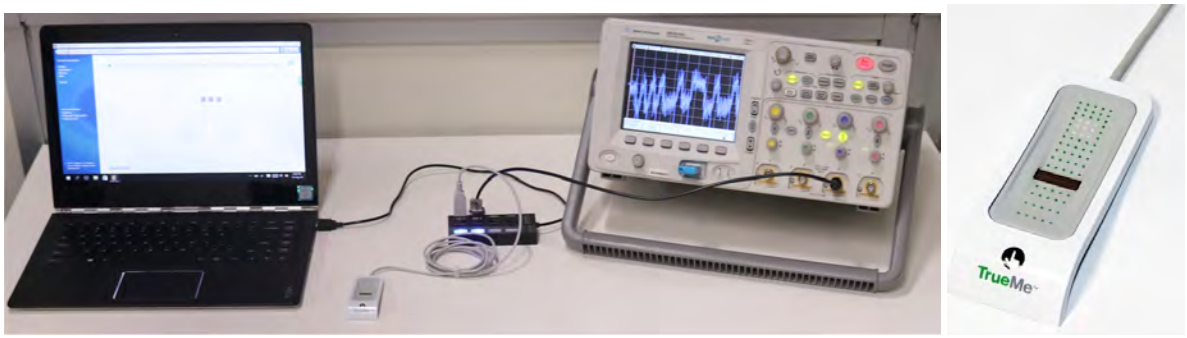


Figure 19: (left) Experimental setup for observing data line crosstalk leakage from a USB fingerprint reader (Eikon Trueme). The data line crosstalk leakage is clearly visible on the oscilloscope's screen. (right) Eikon Trueme fingerprint reader

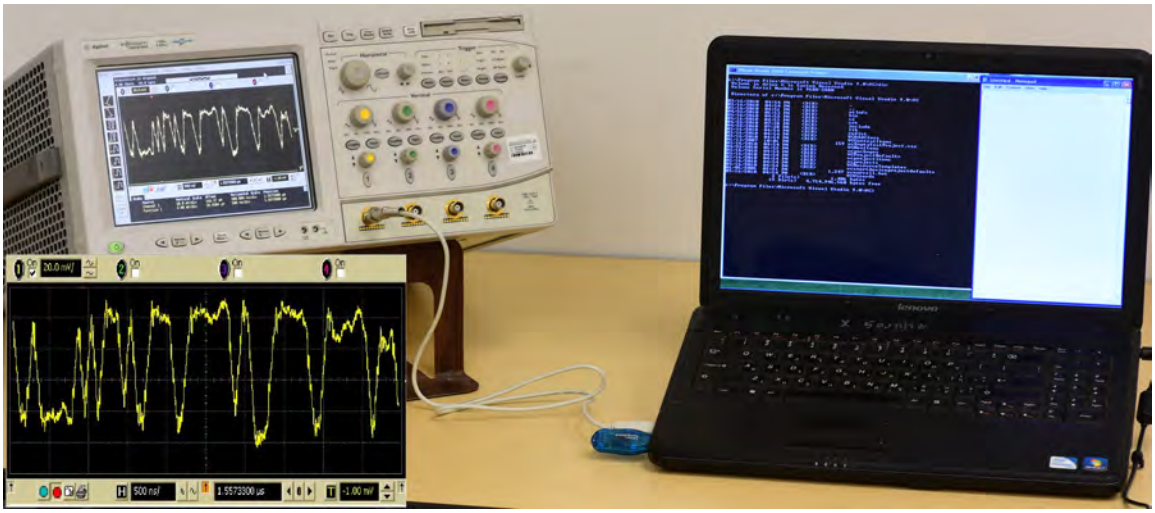


Figure 20: Observing the data line crosstalk leakage during a file transfer from a USB 1.1 drive (blue), connected to the laptop's (Lenovo G550) internal USB 2.0 hub using an Agilent Infiniium DSO 54832B Oscilloscope. The data line crosstalk leakage is clearly visible on the oscilloscope's screen.

We then connected the keyboard, spy probe, mouse and the USB drive via a 4 port USB hub. See Figure 23.

Key Recognition Rate. We measured the spy probe's key recognition rate under various typing speeds. Using a digital metronome as a speed reference, we pressed a random key on every metronome pulse. We evaluated the spy probe's ability to operate at various typing speeds. As can be seen in Figure 24, the spy probe achieves 97%

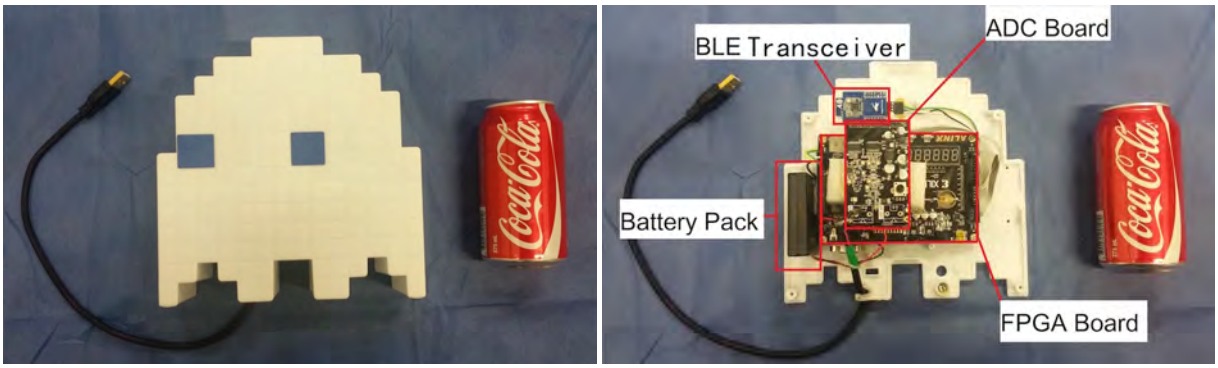


Figure 21: The external appearance of the spy probe, which is embedded inside a toy ghost lamp, size is compared with a 375mL classic Coca-Cola can (left). Inside look of the spy probe, showing the ADC board, FPGA board, BLE board and battery pack (right).

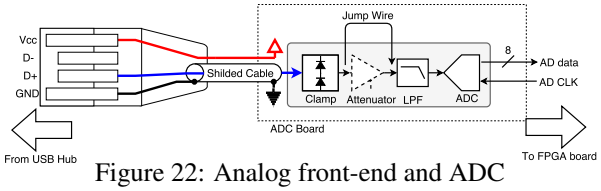


Figure 22: Analog front-end and ADC

accuracy rate for typing speeds from 150 KPM (Keypress Per Minute) to 210 KPM. Notice that average adult typing speed is between 36 and 45 words per minutes, equivalent to 200 KPM.[5]

Figure 23 is a complete demonstration of our attack. We typed "USB CROSSTALK" on the keyboard while the spy probe was monitoring the data line crosstalk leakage, exfiltrating the key presses via bluetooth to the attacker's computer.

7 Conclusions

In this paper we present two attacks on the USB bus, which expose upstream traffic hitherto considered safe against off-path adversaries. The attacks exploit the electrical properties of USB hubs and affect both internal hubs and external hubs. Traditional countermeasures, such as blocking the power or the data lines, do not protect against our attack. We now describe potential countermeasures against the attacks.

Hardware Countermeasures. One possible solution to completely remove any crosstalk leakage is optically decoupling the USB data lines and constructing a dedicated 5V supply for each downstream port. However such solutions are expensive and require careful design. A cheap countermeasure which significantly reduces the power line crosstalk leakage uses an LC low pass filter and LDO (low dropout regulator) to decouple the USB power lines from the data lines. Figure 25 presents an improved USB condom which, in addition to disconnecting the USB data lines, also attempts to suppress any signal above 300Hz. As can be seen in Figure 26, our improved USB condom is able to significantly reduce the data line crosstalk leakage, thus requiring far more sensitive measurement equipment to exploit the small remaining leakage.

Frequency filtering cannot be used to protect the data lines against crosstalk leakage. The leaked signal carries the same basic frequencies as the original signal. Hence any frequency-based filtering that removes the leakage frequencies will also remove the signal frequencies. We leave the problem of designing hardware countermeasures to data line leakage to future work.

Software Countermeasures. The lack of encryption in the USB protocol is a major design limitation of the bus. Without encryption, the design is unable to guarantee the confidentiality and the integrity of messages. Adding end-to-end encryption, for example using the methodology of [7] would protect messages from eavesdropping attacks such as those we describe in this work. Simpler approaches, such as encryption with a session key generated, for example, using the Diffie-Hellman key exchange protocol [20], could also mitigate our attack. Both approaches require devices to have sufficient computational power to perform public key operations.

Future Work. Our spy probe implementation uses commercial off-the-shelf components. Because these are not optimized for the task of capturing USB traffic, they require relatively large space and consume a lot of power. Designing dedicated hardware carries the promise of a small-sized implementation that can be embedded in inconspicuous looking devices and even within the USB plug [40].

While our attack does apply to non-USB 3.0 devices connected to USB 3.0 hubs (see Figure 4(d)), one limitation of our work is that it does not apply to USB 3.0 devices connected to USB 3.0 hubs. This is because USB 3.0 devices connected to USB 3.0 hubs simulta-

[5] http://typefastnow.com/average-typing-speed

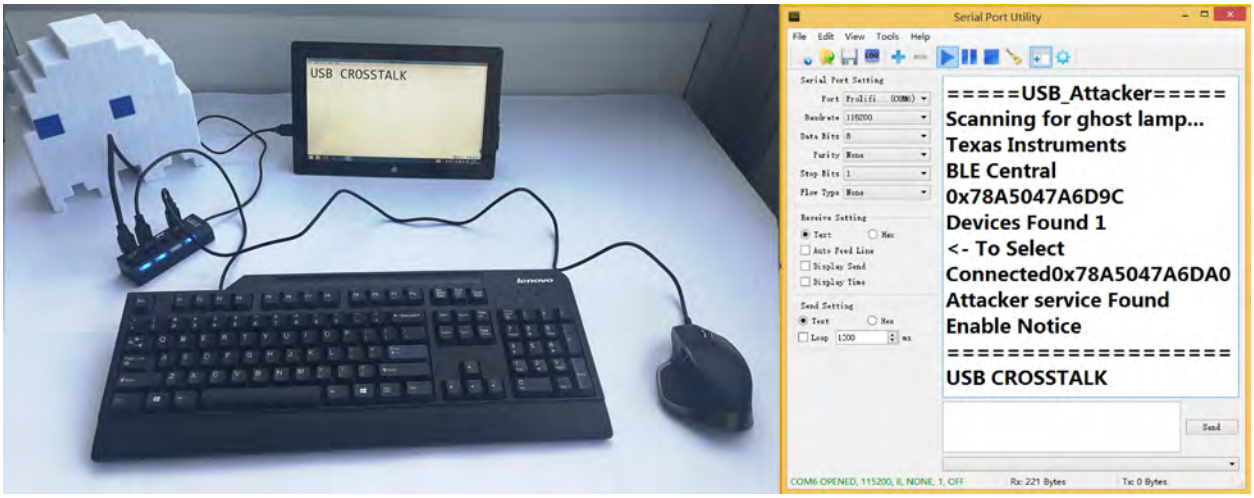


Figure 23: Demonstration of our attack. (left) Phrase being typed on the Surface Pro via a USB keyboard. (right) extracted key presses corresponding to the string "USB CROSSTALK".

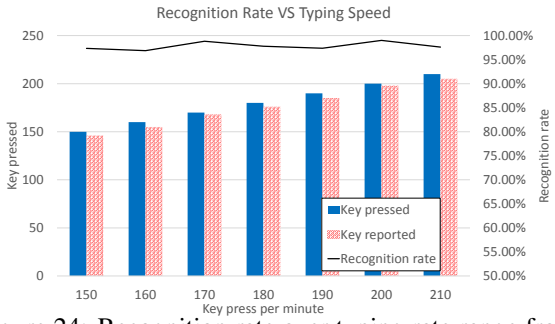


Figure 24: Recognition rate over typing rate range from 150 KPM to 210 KPM.

neously use three differential wire pairs and employ a much higher transmission rate. This configuration significantly exceeds the specifications of our measurement equipment (both in measurement speed and number of channels). While more difficult to attack, USB 3.0 devices connected to USB 3.0 hubs present a lucrative target, in particular because in this configuration the downstream communication, like upstream communication, is unicasted from the host to the device. Exploiting crosstalk effects on such configurations would therefore expose downstream traffic (in addition to upstream traffic) to off-path attackers. As we mentioned earlier, input devices, which often send sensitive information to the host, mostly use USB 1.x Hence, even though our attack does not apply to the newest version of the protocol (USB 3.0), it remains relevant.

The current research applies to USB devices. Further research is required to check if other buses and communication networks are vulnerable to crosstalk attacks.

Acknowledgements

Yuval Yarom performed part of this work as a visiting scholar at the University of Pennsylvania.

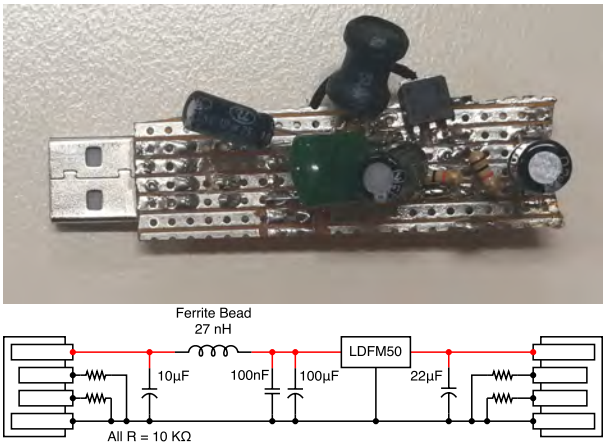


Figure 25: A USB condom which attempts to suppress power line crosstalk leakage.

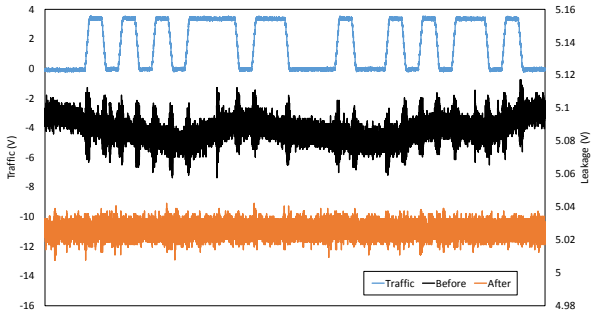


Figure 26: Evaluation of our USB condom showing the actual USB communication (blue), the corresponding unmitigated power line crosstalk leakage (black) and the power line crosstalk leakage measured on the USB port of our USB condom (orange). Notice that the power line crosstalk leakage is significantly attenuated.

This research was supported by an Endeavour Research Fellowship from the Australian Department of Education and Training; the Australian Research Council under project DP140103448; the 2017-2018 Rothschild Postdoctoral Fellowship; the Blavatnik Interdisciplinary Cyber Research Center (ICRC); by the Check Point Institute for Information Security; by the Israeli Centers of Research Excellence I-CORE program (center 4/11); by the Leona M. & Harry B. Helmsley Charitable Trust; the Defence Science and Technology Group, Maritime Division, Australia; the Warren Center for Network and Data Sciences; the financial assistance award 70NANB15H328 from the U.S. Department of Commerce, National Institute of Standards and Technology; and by the Defense Advanced Research Project Agency (DARPA) under Contract #FA8650-16-C-7622.

References

[1] "KeyGhost USB keylogger," http://www.keyghost.com/usb-keylogger.htm.

[2] "KeyGrabber USB," http://www.keelog.com/.

[3] "The original USB Condom," http://int3.cc/products/usbcondoms.

[4] "PortaPow smart charge," http://www.portablepowersupplies.co.uk/portapow-fast-charge-data-block-usb-adaptor/.

[5] "USB Rubber Ducky," http://usbrubberducky.com/.

[6] R. J. Anderson, *Security Engineering: A Guide to Building Dependable Distributed Systems*, 2nd ed. Wiley, 2008.

[7] S. Angel, R. S. Wahby, M. Howald, J. B. Leners, M. Spilo, Z. Sun, A. J. Blumberg, and M. Walfish, "Defending against malicious peripherals with Cinch," in *USENIX Security 2016*, Aug. 2016, pp. 397–414.

[8] D. Asonov and R. Agrawal, "Keyboard acoustic emanations," in *IEEE S&P*, May 2004, pp. 3–14.

[9] M. Backes, M. Dürmuth, S. Gerling, M. Pinkal, and C. Sporleder, "Acoustic side-channel attacks on printers," in *USENIX Security*, Aug. 2010, pp. 307–322.

[10] D. Barrall and D. Dewey, ""Plug and root," the USB key to the kingdom," in *BlackHat 2005*, Jul. 2005.

[11] D. Bertsekas and R. Gallager, *Data Networks*. Prentice-Hall, 1987.

[12] S. Bratus, T. Goodspeed, P. C. Johnson, S. W. Smith, and R. Speers, "Perimeter-crossing buses: a new attack surface for embedded systems," in *Workshop on Embedded Systems Security (WESS)*, 2012.

[13] *Universal Serial Bus Specification, Rev. 1.0*, Compaq, Data Equipment Corporation, IBM PC Company, Intel, Microsoft, NEC and Northern Telecom, Jan. 1996.

[14] *Universal Serial Bus Specification, Rev. 2.0*, Compaq, Hewlett-Packard, Intel, Lucent, Microsoft, NEC and Philips, Apr. 2000.

[15] A. Crenshaw, "Programmable HID USB keystroke dongle: Using the Teensy as a pen testing device," in *DEFCON-18*, Aug. 2010.

[16] ——, "Plug and prey: Malicious USB devices," in *ShmooCon 2011*, Jan. 2011.

[17] *Application Note 83: Fundamentals of RS-232 Serial Communications*, Dallas Semiconductor, 1998.

[18] Dark Purple, "USB Killer," http://kukuruku.co/hub/diy/usb-killer, Mar. 2015.

[19] S. L. Diamond, "A new PC parallel interface standard," *IEEE Micro*, vol. 14, no. 4, Aug. 1994.

[20] W. Diffie and M. E. Hellman, "New directions in cryptography," *IEEE Transactions on Information Theory*, vol. IT-22, no. 6, pp. 644–654, Nov. 1976.

[21] "Thunder missile launcher," http://dreamcheeky.com/thunder-missile-launcher, Dream Cheeky.

[22] R. Frankland, "Side channels, compromising emanations and surveillance: Current and future technologies," Royal Halloway University of London, Tech. Rep. RHUL-MA-2011-07, Mar. 2011.

[23] D. Genkin, L. Pachmanov, I. Pipman, A. Shamir, and E. Tromer, "Physical key extraction attacks on PCs," *CACM*, vol. 59, pp. 70–79, Jun. 2016.

[24] D. Genkin, L. Pachmanov, I. Pipman, E. Tromer, and Y. Yarom, "ECDSA key extraction from mobile devices via nonintrusive physical side channels," in *ACM CCS 2016*. ACM, 2016, pp. 1626–1638.

[25] D. Genkin, I. Pipman, and E. Tromer, "Get your hands off my laptop: Physical side-channel key-extraction attacks on PCs," in *CHES 2014*. Springer, 2014, pp. 242–260.

[26] T. Halevi and N. Saxena, "Keyboard acoustic side channel attacks: exploring realistic and security-sensitive scenarios," *Int. J. Inf. Secur.*, vol. 14, no. 5, pp. 443–456, Oct. 2015.

[27] *Universal Serial Bus 3.0 Specification*, Hewlett-Packard, Intel, Microsoft, NEC, ST-NXP Wireless, Texas Instruments, Nov. 2008.

[28] *High-Definition Multimedia Interface Specification Version 1.3a*, Hitachi, Matsushita, Philips, Silicon Image, Sony, Thomson and Toshiba, Nov. 2006.

[29] *IEC 60130-9: Connectors for frequencies below 3 MHz Part 9: Circular connectors for radio and associated sound equipment, 4th edition*, International Electrotechnical Comission, 2011.

[30] P. C. Kocher, J. Jaffe, B. Jun, and P. Rohatgi, "Introduction to differential power analysis," *JCEN*, vol. 1, no. 1, pp. 5–27, 2011.

[31] M. G. Kuhn, "Electromagnetic eavesdropping risks of flat-panel displays," in *PET 2004*, May 2004, pp. 88–107.

[32] B. Lau, Y. Jang, C. Song, T. Wang, P. H. Chung, and P. Royal, "Mactans: Injecting malware into IOS devices via malicious chargers," in *BlackHat 2013*, Jul. 2013.

[33] J. Maskiewicz, B. Ellis, J. Mouradian, , and H. Shacham, "Mouse trap: Exploiting firmware updates in USB peripherals," in *WOOT'14*, Aug. 2014.

[34] M. Neugschwandtner, A. Beitler, and A. Kurmus, "A transparent defense against USB eavesdropping attacks," in *EuroSec'16*, 2016, pp. 6:1–6:6.

[35] K. Nohl, S. Krißler, and J. Lell, "BadUSB — on accessories that turn evil," in *BlackHat 2014*, Aug. 2014.

[36] J. Oberg, W. Hu, A. Irturk, M. Tiwari, T. Sherwood, and R. Kastner, "Information flow isolation in I2C and USB," in *48th DAC*, Jun. 2011, pp. 254–259.

[37] Y. Oren and A. Shamir, "How not to protect PCs from power analysis," Rump Session, Crypto 2006, 2006.

[38] C. R. Paul, *Introduction to Electromagnetic Compatibility*, 2nd ed. John Wiley & Sons, Inc., 2006.

[39] J. Rutkowska, "Evil Maid goes after TrueCrypt!" http://theinvisiblethings.blogspot.com.au/2009/10/evil-maid-goes-after-truecrypt.html, Oct. 2009.

[40] B. Schneier, "COTTONMOUTH-I: NSA exploit of the day," https://www.schneier.com/blog/archives/2014/03/cottonmouth-i_n.html, 2014.

[41] M. Schulz, P. Klapper, M. Hollick, and E. Tews, "Trust the wire, they always told me!: On practical non-destructive wire-tap attacks against Ethernet," in *WiSec'16*, Jul. 2016, pp. 43–48.

[42] R. Sevinsky, "Funderbolt: Adventures in Thunderbolt DMA attacks," in *BlackHat 2013*, Jul. 2013.

[43] *External Serial ATA*, Silicon Image, Sep. 2004.

[44] F. Steinmetz, "USB — an attack surface of emerging importance," Bachelor Thesis, Hamburg University of Technology, Mar. 2015.

[45] D. Tian, N. Scaife, A. Bates, K. R. B. Butler, and P. Traynor, "Making USB great again with USBFILTER," in *USENIX Security 2016*, Aug. 2016, pp. 415–430.

[46] J. D. Tian, A. M. Bates, and K. R. B. Butler, "Defending against malicious USB firmware with GoodUSB," in *Proceedings of the 31st Annual Computer Security Applications Conference, Los Angeles, CA, USA, December 7-11, 2015*. ACM, 2015, pp. 261–270.

[47] *VESA DisplayPort Standard Version 1, Revision 1a*, Video Electronics Standards Association, Jan. 2008.

[48] M. Vuagnoux and S. Pasini, "Compromising electromagnetic emanations of wired and wireless keyboards." in *USENIX security 2009*, 2009, pp. 1–16.

[49] L. Wang and B. Yu, "Analysis and measurement on the electromagnetic compromising emanations of computer keyboards," in *CIS 2011*, Dec. 2011, pp. 640–643.

[50] C. Wisniewski, "Windows zero-day vulnerability uses shortcut files on USB," https://nakedsecurity.sophos.com/2010/07/15/windows-day-vulnerability-shortcut-files-usb/, Jul. 2010.

[51] Q. Yang, P. Gasti, G. Zhou, A. Farajidavar, and K. Balagani, "On inferring browsing activity on smartphones via USB power analysis side-channel," *IEEE Transactions on Information Forensics and Security*, no. 99, pp. 1–1, 2016.

[52] J. Zaddach, A. Kurmus, D. Balzarotti, E.-O. Blass, A. Francillon, T. Goodspeed, M. Gupta, and I. Koltsidas, "Implementation and implications of a stealth hard-drive backdoor," in *29th ACSAC*, Dec. 2013, pp. 279–288.