
Department Informatik

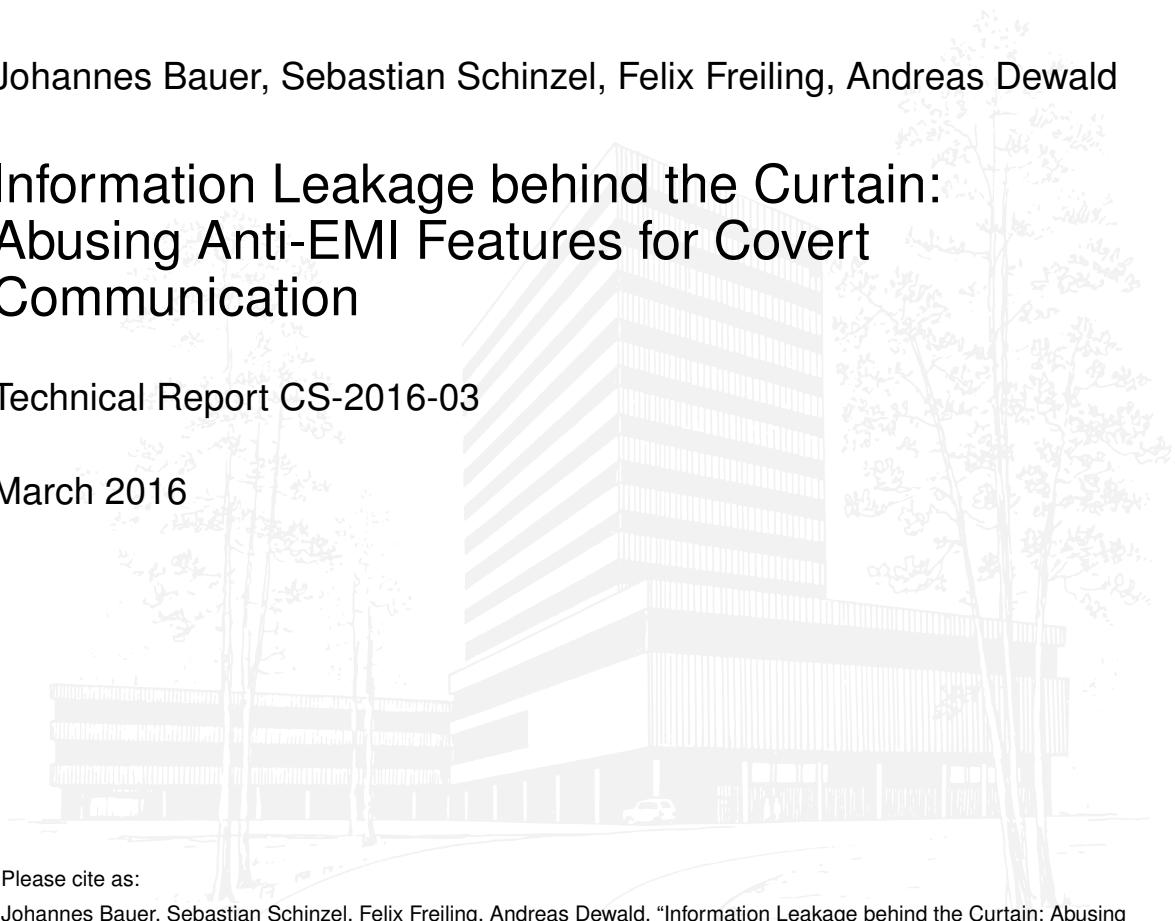
Technical Reports / ISSN 2191-5008

Johannes Bauer, Sebastian Schinzel, Felix Freiling, Andreas Dewald

Information Leakage behind the Curtain: Abusing Anti-EMI Features for Covert Communication

Technical Report CS-2016-03

March 2016



Please cite as:

Johannes Bauer, Sebastian Schinzel, Felix Freiling, Andreas Dewald, "Information Leakage behind the Curtain: Abusing Anti-EMI Features for Covert Communication," Friedrich-Alexander-Universität Erlangen-Nürnberg, Dept. of Computer Science, Technical Reports, CS-2016-03, March 2016.

Information Leakage behind the Curtain: Abusing Anti-EMI Features for Covert Communication

Johannes Bauer^{*†}, Sebastian Schinzel[‡], Felix Freiling[†], Andreas Dewald[§]

^{*}Robert Bosch Smart Home GmbH, Stuttgart-Vaihingen

[†]Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU)

[‡]Münster University of Applied Sciences

[§]ERNW Research GmbH, Heidelberg

Abstract—We present a new class of covert channels which can be created by utilizing common hardware but that cannot be detected by such. Our idea is to abuse anti-EMI features of a processor to create a covert channel on the physical layer. Thus, the sender uses the invariants in how digital signals are encoded over analog channels to covertly transport information. This leaked data is present on the wire bound connections of the compromised device, but is also by definition present in the vicinity of the device and can be picked up by radio equipment. As the covert channel is present only on the physical layer, the data on all layers above, as well as the timing behavior on those layers is indistinguishable from uncompromised devices. We present two example implementations of such channels using RS-232 as the carrier and use a common oscilloscope to decode the resulting covert channel. Using this setup, we observed symbol rates of around 5 baud. We derive the theoretical upper bound of the covert channels bandwidth and discuss the factors by which it is influenced.

I. INTRODUCTION

On the lowest layer of the OSI model, data is transmitted over a physical medium like a wire. In order to do this, the source data is encoded into physical parameters of the medium such as voltages or currents and thus comprises a signal $f(t)$. Any physical medium is subject to *noise* which can be modeled as an additive component to f . This means that the receiver will not receive the pure signal $f(t)$ but rather $f(t) + g(t)$ where g models the noise. Tolerance to noise on the physical layer is achieved by certain tolerance levels implemented by the interpretation function at the receiver. So, if \mathcal{D} denotes the function which translates the analog signal on the physical layer to code words on the data link layer, this function satisfies the following equation for error-free transmission:

$$\mathcal{D}(f(t) + g(t)) = \mathcal{D}(f(t)) \quad (1)$$

As an example, consider the transmission of a single byte using RS-232. Note that we selected RS-232 because the simplicity of the protocol allows for a precise presentation of our core ideas. Protocols like USB, SPI or I²C would work equally well. In Fig. 1, ten RS-232 symbols are transmitted on the physical wire: one start bit, 8 data bits and one stop bit. Since the bit order is LSB-first, in the example the value 0xb1 is

transmitted. Instead of letting involuntary noise $g(t)$ act upon the signal we could similarly specifically craft a function that would slightly change the signal in a way that would not alter the outcome of the interpretation function \mathcal{D} . Examples for this could be marginally lower or higher voltages, slightly faster or slower transitions between the low and high states or a small phase shift of the signal. Two of these examples are shown in Fig. 2. The signal is sampled at the indicated points in time. The variations of the transition speed or signal phase do not matter on the digital layer as long as the signal value for the bit value 0 would stay below the input low threshold V_{IL} and as long as it would stay above V_{IH} for the value 1 at the sampling

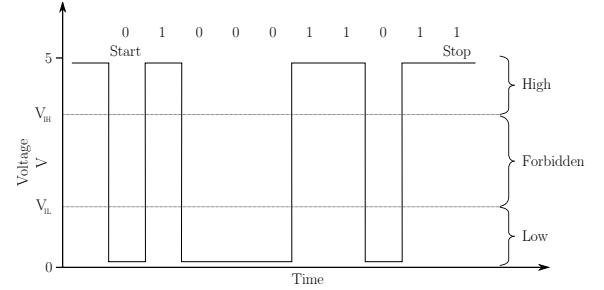


Fig. 1. Exemplary RS-232 transmission of a single octet

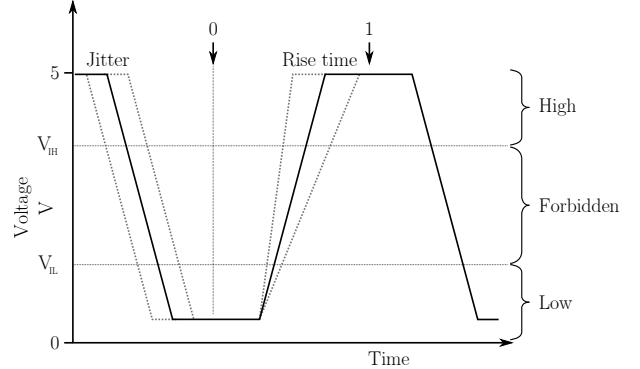


Fig. 2. Signal changes of jitter/rise time that preserve the physical property

points. The interpretation \mathcal{D} is guaranteed to remain identical and thus the physical property is preserved.

The observation we make in this paper is the following: If $g(t)$ is misused to encode secret information by slight variations in voltage or timing while ensuring that $\mathcal{D}(f(t) + g(t)) = \mathcal{D}(f(t))$, then there is no easy way for the standard receiver to decode or even detect this information. However, if $g(t)$ can be measured by a *specialized receiver* with an interpretation function \mathcal{D}' (such as an oscilloscope with custom recovery algorithms), it is possible to extract the information while from a data link point of view, there is no observable difference between the modified and unmodified signals. An attacker who knows the exact signal deviations caused by the implanted covert channel could easily build a specialized receiver with dedicated hardware. Such hardware could be an FPGA development board for which the total hardware cost of the receiver would be somewhere in the range of around \$100.

A. Attacker Scenario: Covert Communication

Consider an attacker who wishes to covertly access secrets that are inserted into sensitive security hardware (like a tamper-resistant key store) *after* it has been deployed. In order to achieve his goal, the attacker acquires access to the supply chain between a silicon manufacturer and the original equipment manufacturer (OEM), as illustrated in Fig. 3. Within the supply chain the attacker is able to intercept and modify the hardware, and later can get *close* to the hardware in the field to access the secrets without actually having physical access to it.

Clearly, an attacker with physical access to a device has many possibilities to create backdoors. We however assume that (1) the attacker can only physically access and modify the device once, and (2) the modified devices are subject to intensive security checks by the OEM before deployment. Therefore, covertness cannot be achieved by classical hardware backdoors [15] and the extraction of information from hardware by physical access [10, 11] is not an option. Note that such attack scenarios are not uncommon in practice [1], and in common end-of-line test bedding environment scenarios run by OEMs (such as a bed of nails test fixture) the focus is only on the *digital* semantic correctness of the devices under test. A covert channel in the way we describe in this paper would pass such a digital test effortlessly, even when probed for with more sophisticated methods like fuzzing.

We demonstrate that the logic that is necessary to perform such an attack is minimal – in fact, many modern devices already have abundance of possible circuitry on board which would allow deployment of such a channel. To demonstrate that little hardware modification is needed we show that the already present circuitry in off-the-shelf hardware is completely sufficient to construct a covert channel with it by doing only modifications of software (i.e., the firmware).

B. Abuse of Anti-EMI Features

In this paper, we show that anti-EMI functionality can be misused to implement a covert channel and want to

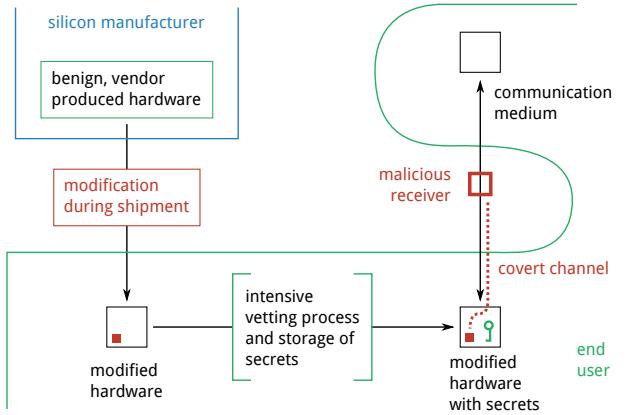


Fig. 3. Example of supply chain poisoning

raise awareness of such threats. *Electromagnetic interference* (EMI) is an unwanted and inconvenient side effect which every electronic device exhibits. Governmental regulations limit the maximum amount of emitted EMI and so the signal processing logic of many electronic devices contains suppressing techniques such as *Spread Spectrum Clocking* (SSC) or *Rise Time Control* to reduce the EMI emission. Our idea is to use such facilities as a covert communication medium. Both options are available on standard microprocessors today and can be used by software to reduce radiated emission. This allows us to create a covert channel with the following properties:

- 1) because it can be realized in software, *sending* information on the channel is easy and can be performed with a wide range of commercial off-the-shelf hardware, and
- 2) *receiving* and decoding the information requires specialized measurement equipment such as oscilloscopes or custom hardware. This renders the channel invisible to an observer on the data link layer.

C. Related Work

The notion of covert channels goes back to Lampson in 1973 [16] when he distinguished *timing channels* and *resource channels*. Timing channels encode information in the inter-packet timing delay, while resource channels use packet ordering or the state of a packet to transport information. Later, Kemmerer [14] generalized the notion to scenarios with any form of shared resources. While the concept evolved in military contexts, today, the threats of covert communication have reached the mainstream and various implementations based on many different communication methods, such as IP, exist as research prototypes [9, 20, 25] and “in the wild” [1, 5]. While the specific encoding and methods for creating covert channels were refined, the actual implementation almost always focuses on protocol layers at or above the data link layer [13, 17, 21, 31]. Consequently defensive methods, i.e., attempts to detect a covert channel, usually assume a packet abstraction such as the one provided by the Internet Protocol [4, 8, 18, 33]. In contrast, our work is completely independent of such an abstraction.

There also exists some work that makes use of physical properties of hardware or communication on the data link layer to implement covert communication [6, 12, 15, 27, 30]. For example, work on hardware Trojans falls into this category, e.g., King et al. [15], Tehranipoor and Koushanfar [30], Farag, Lerner, and Patterson [6]. They embed malicious circuitry in FPGA targets and therefore augment the present hardware in order to create backdoors. In contrast, our approach can be implemented using functionality that is already present in MCUs and can often be achieved by only modifying the firmware. Moreover, existing work is detectable by observing the digital behavior of the circuit.

Work by Iakymchuk et al. [12] that uses heat dissipation to implement a covert channel can be regarded very close to our work, but their proof of concept implementation also requires hardware modification by altering the FPGA netlist. Shah and Blaze [27] use the physical properties of the transport medium to implement a covert channel and encode covert information by selectively disrupting the physical carrier. Their attack however requires special radio frequency equipment, i.e., specialized sending and receiving hardware since a deliberate carrier disturbance is not possible with benign hardware.

Interesting observations are presented by Genkin et al. [7]; they analyzed low-cost methods of key recovery in systems which exhibit electromagnetic side channel emission by using a Software Defined Radio (SDR). Their recovery approach could be applied to recovery of our EM covert channel as well.

D. Contributions

In this paper, which is the extended version of the presentation at HOST 2016 [2], we make the following contributions:

- We introduce a new class of covert channels that uses *sub-digital* means to transport information, i.e., it abuses the degrees of freedom in the representation of digital signals on physical channels. In a sense, instead of looking at the *inter-packet* delay, our covert channels modify timing properties *within* packets, i.e., we use *intra-packet* timing channels. To the best of our knowledge, we are not aware of any other work that formulates and demonstrates this idea.
- We argue that these channels pose a relevant threat by showing that they can be *implemented* easily in *software*. We demonstrate this by using anti-EMI features that are supported by many commercial off-the-shelf processors. Such channels have an *asymmetry* property in that it is easy to send information over the channel but it requires special hardware to decode the covertly transmitted information. This in turn means that deliberate, targeted effort is required that specifically looks at aspects of the signal in order to detect the presence of such a covert channel. Our proof-of-concept example implementation demonstrates two different ways to encode information on RS-232 as the carrier protocol.

Since we exploit sub-digital features, the attacker necessarily needs either physical access to the compromised medium over which the information is sent or close physical proximity to

the device. This is because by definition our covert channel is present only in the analog/digital encoding ambiguity and as a side effect in the form of parasitic electromagnetic emission. A standard receiver or relay (for example a network switch in the case of Ethernet) will destroy the covert signal because it digitally interprets and reconstructs passed on signals. This means an attacker has to deploy a decoding unit somewhere within the network (for example by intercepting the wire) or use radio frequency equipment in the vicinity of the compromised device. We argue that this is a necessary downside of this new type of covert channel.

Note that for our example implementation we chose RS-232 for convenience only. Our approach could likewise be applied to many other carrier protocols, including but not limited to I²C, SPI, I²S or USB [19, 23, 24]. Even though RS-232 has largely disappeared from the desktop computing environment, it is still widely used in embedded environments as a means of chip-to-chip communication. The fact that the output driver configuration is independent of the selected function of the MCUs port pin means that all peripheral functions on that port pin are affected by our channel. Our second approach affects the main system clock of the microcontroller (and because all peripheral clocks are derived from that clock source) directly leads to the consequence that *all* output peripherals are affected in exactly the same manner as the RS-232 transport. This includes all peripherals that are supported by the used MCU. An attacker only needs to be able to monitor at least one affected channel in order to recover the covertly transmitted data. That these properties are independent of the used communication protocol makes our approach very versatile.

E. Outline

This paper is structured as follows: In Sect. II, we discuss the signal theoretical background information necessary to understand the used carrier signal and also describe the anti-EMI mechanisms that modern microcontrollers employ. We continue by demonstrating two concrete covert channels in Sects. III-A and III-B which we implement with a Cortex-M4 microcontroller [28]. Section IV gives a brief overview of how the transmitted symbols are encoded in our case and elaborates on the theoretic maximum channel capacity. Then we continue to explain how our channel can be applied to a real-world scenario in Sect. V. Finally, Sect. VI gives a summary and an outlook on what future work could be based on these methods.

II. BACKGROUND: ELECTROMAGNETIC INTERFERENCE

Digital square wave signals are composed of superimposed sine waves of different frequencies and amplitudes. Any square wave signal can be decomposed into its components by means of the Fourier transformation [3]. For a square wave with n harmonics, i.e., n superimposed integer multiples of the fundamental frequency, the signal amplitude at a point in time φ is given by

$$f(\varphi) = \sum_{i=0}^n \frac{1}{2i+1} \sin((2i+1)\varphi). \quad (2)$$

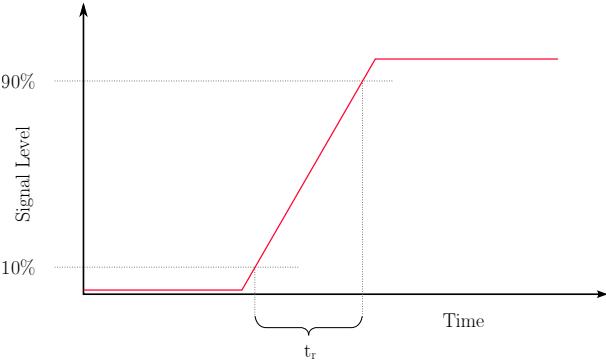


Fig. 4. Illustrated rise time of a signal

Note that the second derivative of this function $d\varphi$ is

$$f''(\varphi) = - \sum_{i=0}^n (2i+1) \sin((2i+1)\varphi). \quad (3)$$

By solving $f'' = 0$, one can see that the function has an inflection point at $\varphi = 0$, which yields $f'(0) = n + 1$. This means, the maximum slope of the composed digital signal, $n + 1$, is directly related to the number of contained harmonics. Thus, an ideal square wave signal has a positive edge slope of infinity and consequently contains an infinite number of harmonics.

Real-world digital signals cannot change instantaneously and the transit from the *low* to the *high* state or vice versa takes a certain amount of time. This time is referred to as the *rise time* when a signal does a low to high transition and *fall time* in the inverse case. Since the following argumentation applies to both rise and fall times analogously, we only discuss rise times in detail. By convention, the measured rise time begins when the signal has reached 10% of its peak value and ends when it is at the 90% mark. This is illustrated in Fig. 4.

The edge that constitutes the low to high transition also has a certain *slope*, usually measured in voltage per unit of time. A steeper slope intuitively corresponds to a shorter rise time and vice versa. We refer to both during this paper interchangeably. The steeper the slope of a digital signal is, the more harmonics are contained within the signal. Additionally, part of the dissipated power of signals is emitted in the form of *electromagnetic interference* (EMI). Intuitively spoken, this means that the device involuntarily acts as a radio transmitter. This effect becomes especially significant at high frequencies. It is therefore advantageous to limit the slope of the signal to the amount that is necessary by the constraints of the connected peripheral. Any signal with steeper slope only increases EMI though the presence of additional high-frequency harmonics, but will not benefit the actual data transmission.

Apart from the actual rise time of the signal, two other factors influence the emission spectrum of any device: One is the clock frequency which also directly affects the frequencies and amplitudes of all contained harmonics. The other is the antenna efficiency of the parasitic antenna that is constituted by the integrated circuit itself. Antenna efficiency is highly

nonlinear over the frequency spectrum. In order to reduce unwanted electromagnetic emission, one could theoretically change any of these factors. While changing the characteristic of the parasitic antenna is not possible in software, changing the clock frequency easily is. The main idea of *spread spectrum clocking* is therefore to vary the clock frequency periodically in order to *smear* the emitted spectrum. Although this means that the emitted energy (i.e., the integral of the power over the frequency spectrum) stays nearly the same, spectral peaks which could cause trouble in an EMI examination can easily be avoided, as Fig. 5 shows.

For spread spectrum clocking, there are two parameters that influence how the clock frequency f changes over time: The period at which a frequency is modulated is called P and the associated modulation frequency is called f_m . The amplitude of the modulation is referred to as the *modulation depth* and is abbreviated with d . The example in Fig. 6 uses triangular clock modulation.

Microcontroller manufacturers are aware of EMI problems and have therefore equipped many of the newer devices with the possibility to limit the rise and fall times of digital signals by use of special configuration registers [28]. According to the needs of the application, the analog properties of the signal can be modified within certain limits. Even with enabled memory protection by means of the MPU, in many scenarios at least some of these registers (such as the GPIO risetime configuration) will be accessible to unprivileged application software.

For implementing our covert channels, we chose a general-purpose microcontroller of the ARM Cortex-M family, the STM32F407VG [28]. Chips within that family are priced starting from \$2 up to around \$20, depending on the amount

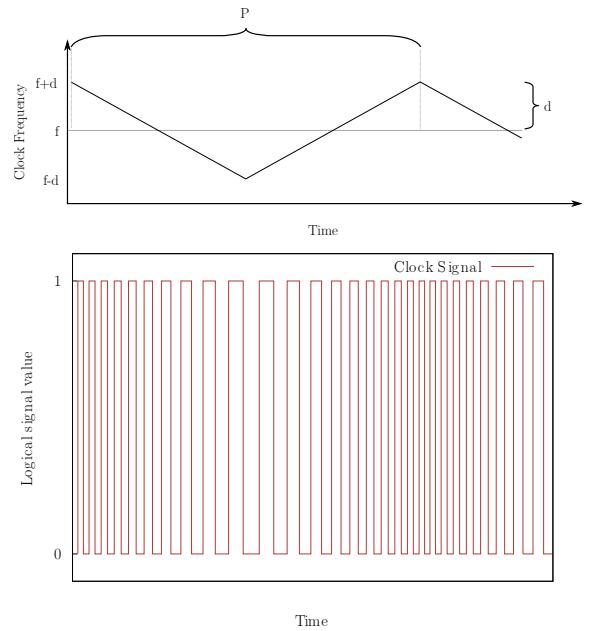


Fig. 6. Frequency of a spread-spectrum clock signal

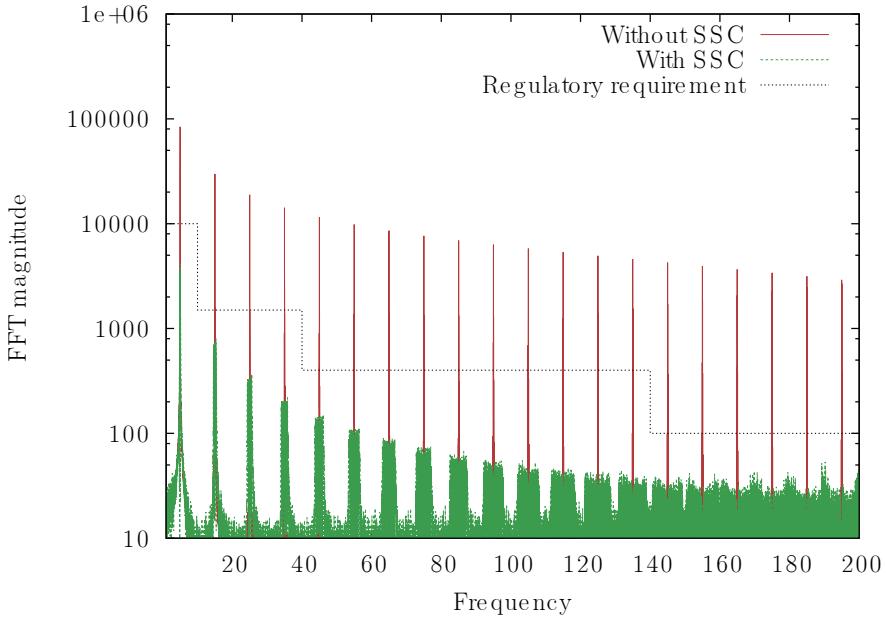


Fig. 5. FFT magnitude of a clock signal with and without Spread Spectrum Clocking and exemplary regulatory permissible maximum to pass EMI regulations

of peripherals they are equipped with. The STM32F407 is approximately the middle of that price range. They are built into a variety of embedded devices such as automotive ECUs (Electronic Control Units), RFID readers and consumer appliances like wireless routers, printers or air conditioners [26].

III. IMPLEMENTATION OF THE COVERT CHANNEL

The anti-EMI facilities that microcontrollers provide are usually configurable in software. An application programmer must be able to decide how and when these mechanisms are enabled since some may have a detrimental effect on the overall performance of the system. For example, while a certain anti-EMI measure may improve electromagnetic emission, it could also simultaneously have a negative effect on the sampling precision of an analog-digital converter (ADC). Therefore, it makes sense to give the application programmer the ability to turn the anti-EMI features on and off at will. This means the chip itself is equipped with the functionality to directly influence how much EMI is emitted at any point in time — a fact that we exploit in the following sections in order to construct a covert channel.

A. Implementation using Spread Spectrum Clocking

We now show how to implement a covert channel using spread spectrum clocking (SSC). While SSC can arguably be implemented in lots of ways, the devices we looked at (the ST32F4xx family) provide means to modulate the clock with a triangle signal of up to 10 kHz and a modulation depth d of 0.25% to a maximum of 2% [29]. Either of both variables can be used to encode data. In order to achieve covertness, it is beneficial to choose the parameters so that the resulting signals look like they have been affected by naturally occurring clock jitter. For comparison, the STM32F4xx datasheet [29] lists a

typical peak-to-peak period jitter of $\pm 200\text{ps}$ which is always present while at 168 MHz a worst-case 2% SSC modulation depth corresponds to a period anomaly of only around 119ps.

In our experiments, we used both variants: modulating the SSC modulation depth on one hand and modulating the modulation frequency on the other. For our experiments we never changed both variables at once, but always kept one of the two constant. Therefore, if modulation depth was modulated, the modulation frequency was chosen to be fixed at 10 kHz. For data transmission using a variable modulation frequency, the modulation depth was constant at 0.25%.

One drawback of choosing the SSC unit on the ST32F4xx family is that each change between states requires the internal phase-locked loop (PLL) to be shut off before the CPU will allow modification of the SSC registers [28]. This is a comparatively lengthy procedure and takes approximately $170\mu\text{s}$ in our case. Shutting down the PLL requires the system clock to be switched to a different clock source; typically this will be the much slower internal RC oscillator. This is an operation which would appear suspicious to someone monitoring a continuous stream of output data with an oscilloscope or similar measurement equipment.

The advantage of using SSC is that the covert information is encoded in what resembles ordinary jitter. Here, d directly corresponds to the amplitude of artificially generated jitter while f_m corresponds to the rate at which the jitter amplitude changes.

Recovery was performed with a Agilent DSO-X 3014A oscilloscope. We triggered on a rising edge of the carrier and dislocated the trigger point one bit length in time, effectively showing the jitter which was artificially generated by the SSC. This data was transmitted to a PC using the USBTMC

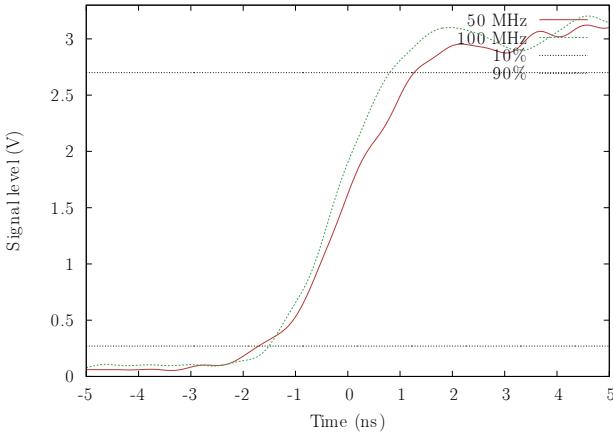


Fig. 7. Signal differences of 50 MHz and 100 MHz drivers

(USB Test and Measurement Class) protocol. The achieved covert symbol rate was about 2 baud using constant 10 kHz modulation frequency and three distinct modulation depths (i.e., three distinct symbols).

The limiting factor for this type of covert channel is the long fixed time which is necessary by hardware constraints to switch from one state to another. For each such change, the PLL has to be stopped and restarted, which takes a fixed amount of time. Therefore, in order to covertly use the SSC unit in order to transmit information, the transmission speed (i.e. number of state changes per unit of time) would have to be exceptionally low so that the time for the state change itself becomes negligible.

B. Implementation using Rise Time Control

We now show how to implement a covert channel using a completely different method than SSC, namely the rise time control of the microcontroller unit (MCU).

The STM32F407VG which we used provides a facility to modify the output speed of the general purpose input/output (GPIO) pins in four different speed categories: 2 MHz, 25 MHz, 50 MHz and 100 MHz. Of those four, the measured average rise times were 19ns, 4.3ns, 2.4ns and 2ns at $V_{cc} = 3.3V$. While three of these (19ns, 4.3ns, 2ns) are trivially distinguishable from each other we wanted to stress covertness of our channel. This is why we chose to only select the 2.4ns and 2ns alternatives even though discriminating those options is technically most challenging.

Note that the selection of output driver speed does *not* affect the transmitted overt bit rate in any way; it only affects the slew rate of the signal and therefore the theoretically maximally achievable bandwidth using that output driver. For example, Full Speed USB uses a 12 MBit/s data channel. For this type of communication either one of the 25, 50 or 100 MHz drivers could be used with no observable difference in the overt communication. A plot that highlights the subtle differences in rise time is shown in Fig. 7. It was captured using a Tektronix MSO4034.

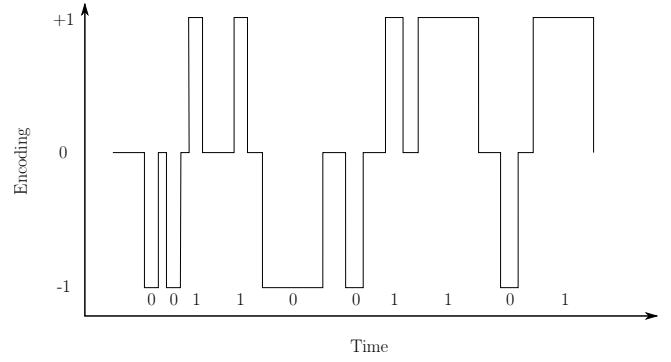


Fig. 8. Ternary encoding of data bits with varying data clock rate

This channel has the advantage that state changes are exceptionally fast in software and that access to the required GPIO registers will usually be allowed even to unprivileged software. That is, even when the chip's MPU is used, access to the relevant memory regions will usually be permitted even to user space applications. Since the difference between the two fastest output driver states is so marginal, the channel also exhibits an outstanding covertness property. Furthermore, it is versatile in the sense that it can be applied to any output which does not require more than 50 MHz of signal bandwidth; it is therefore applicable to a wide variety of output peripherals (e.g. USB, I²C, SPI, etc.).

IV. DATA ENCODING

Depending on the number of discrete states that a receiver can discriminate, an appropriate encoding can be chosen for transmission of data. Concretely, we used a ternary encoding for the SSC variant described in Sect. III-A and a binary encoding for the rise time approach of Sect. III-B. Note that the choice of encoding is orthogonal to the type of channel (i.e., SSC channel or rise time channel) itself.

The ternary variant allows for trivial clock recovery: With symbols $-1, 0, +1$, data bits are encoded by the transitions between the *idle* state 0 to and from either the -1 or $+1$ state. We furthermore require that only the transitions $-1 \leftrightarrow 0$ and $+1 \leftrightarrow 0$ are valid and all other symbol transitions constitute an invalid encoding.

Fig. 8 shows how this encoding looks in practice. Clock recovery is trivial: the only restriction is that the frequency of transmitted bits may not exceed the Nyquist frequency of the recovery unit's sampling rate—in other words, all three symbols must at least appear long enough to be reliably detected but they can appear arbitrarily longer.

When using binary encoding, the clock recovery of the demodulated covert channel is more complex than with a ternary approach. This is due to the fact that the actual data that is transmitted is intermixed with the data clock, i.e., the information at which point in time the data is valid. We relax the difficulty by assuming that the malicious code within the system is called at constant intervals, providing at least clock stability (yet at unknown frequency) and that transmitted data

	SSC	Rise time
Symbol Count (n)	3	2
Switch Time (t)	170 μ s	120 ns
Overt symbol rate (b_o)	10 kBd	115.2 kBd
Covert symbol rate (b_c)	10 kBd	75.9 kBd

TABLE I
EXAMPLES OF CHANNEL PARAMETERS FOR OUR CASE

is random. This is not an unreasonable assumption because leaked data will usually be cryptographic material. Even with completely random data, however, there is often significant disparity within the signal which complicates clock recovery. In order to avoid this, a bit stuffing technique like 8B10B encoding could be used to keep signal disparity to a minimum [32]. Alternatively whitening of the signal using a LFSR-based synchronous additive scrambler could be used. The associated cost would in both cases be a significantly increased malicious code size, which is why we did not explore this further and assumed our transmission signal to already be without relevant bit-bias. Randomly generated cryptographic keys should exhibit no significant bit-bias. An example of the actual measurement data that we did this type of clock recovery on can be seen in Fig. 9.

In summary, with these encodings, we achieved recovery speeds of about 1 bit/sec for the SSC variant (symbol rate of 2 baud, ternary encoding) and about 2.5 bit/sec for the GPIO variant (symbol rate of 5 symbols/sec, binary encoding).

A. Channel Capacity

After measuring the actual transmission speeds of our implementation, we now investigate the theoretically achievable maximum channel capacity. To calculate this, there are three main variables that have to be taken into account:

- 1) Time it takes to switch between one output symbol to another (t)
- 2) Number of distinct unique covert output symbols (n)
- 3) Effective baud rate b_c in dependence on the overt channel baud rate b_o

The two channels that we demonstrate experimentally in Sect. III-A and III-B use Spread Spectrum Clocking (SSC) on one hand and rise time encoding on the other hand as the covert transport. In the SSC example we modulated the modulation depth d to create the channel. For the channels that we create, an overview of the values of these constants is shown in Tab. I. The symbol switch time t is significantly greater in the SSC variant compared to the rise time variant due to the necessity of stopping the phase locked loop (PLL) in order to change the SSC registers of the ST32F407, as already explained in Sect. III-A. It has also been explained there that the covert symbol rate depends not only on the overt symbol rate, but also on the transmitted data. For our examples we transmitted alphanumeric protocol data which gave us the shown values for b_c .

The reason why there is data dependence of b_c on b_o can intuitively explained by the fact that the number of edges within

the signal – and therefore the number of possibilities to inject covert data – depends on the data. For a worst-case word of 0x00 the number of symbols is minimal (two edges per byte of data) while for a constant stream of 0x55 it is maximal (10 edges per byte of data). The plain text we transmitted had on average 6.6 edges per transmitted byte of data.

For our calculations and in order to get a theoretical upper bound, we assume the best case of having at least one edge change per carrier symbol transmission. We then can derive the maximum channel capacity symbol rate as

$$B = \frac{\log_2 n}{t + b_c^{-1}}$$

We provide exemplary calculation for the parameters chosen in our actual experiments and we give an estimate for the theoretical maximum capacity in Tab. II. In the experiments we send alphanumeric data over the RS-232 overt channel with an average of approximately 6.6 edges per transmitted byte, accounting for the lower b_c . For easy discriminability we also limited the number of used symbols n significantly from the theoretical maximum.

For our channel, the maximum b_c would be equal to b_o , i.e. 115.2 kBd, and we have 4 discrete symbols available. Therefore, we could achieve $B = 227.3$ kBd. On first glance, it is counter-intuitive that the covert channel capacity could ever exceed the overt channel capacity. This has several reasons:

- 1) The symbol count of the covert channel can exceed those of the overt channel.
- 2) When the covert symbol rate depends on the transmitted data, we assume the best-case values. In our example this means that a constant overt data stream of 0x55 would need to be sent—something that does not make sense in the real world.
- 3) Any computational power that is needed to control the covert channel is neglected.

In conclusion, while the constructed covert channel might theoretically have a large bandwidth, there are many practical aspects which decrease the practically achievable bandwidth by about five to six degrees of magnitude. Data that an attacker would want to leak through such a channel is in all likelihood cryptographic material that is exceptionally short. Therefore the drawback of limited practical channel capacity is outweighed by the advantage that the channel itself is difficult to detect.

For our experiments the factor that by far dominated the achieved covert bandwidth was not the transmitter, but the

Variable	SSC		Rise Time	
	Experiment	Theory	Experiment	Theory
n	3	193	2	4
t	170 μ s	170 μ s	120 ns	120 ns
b_c	10 kBd	10 kBd	75.9 kBd	115.2 kBd
B	5.9 kBd	28.1 kBd	75.2 kBd	227.3 kBd

TABLE II
THEORETICALLY ACHIEVABLE B UNDER IDEAL CONDITIONS COMPARED TO CONDUCTED EXPERIMENTAL EVALUATION

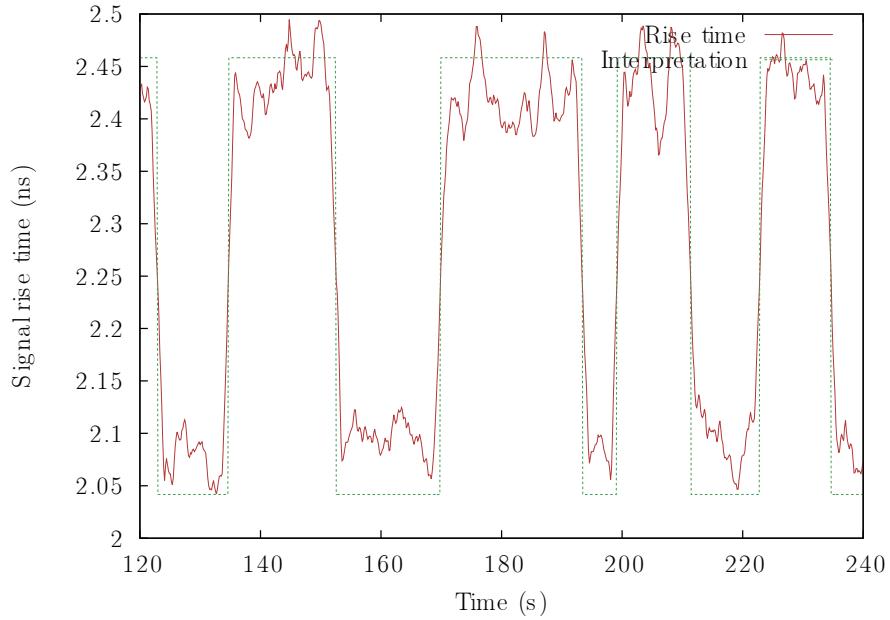


Fig. 9. Rise time over time and the conditioned signal

receiver. Since we relied on general-purpose equipment, the recovery bottleneck was the USBTMC transmission of the results to the decoding PC.

V. PRACTICAL EXAMPLE

We now elaborate in depth on how an attack as motivated in Sect. I-A could look like and how it could be applied in a real-world example. As we stated before, our type of covert channel can be applied to every scenario where the output can be modulated in ways that are invariant for the digital interpretation function. The channels we talked about in-depth previously were wirebound channels which misused the ability to influence involuntary electromagnetic emission.

Our idea can be explored further, however. In the realm of analog circuitry it is quite common that fine-tuning of antenna circuits is done in software. For vendors of radio frequency equipment, this is rather convenient, since it enables them to fine-tune antennas to their circuits using a completely automatic process where the alternative would otherwise be the manual tuning of hardware components such as variable capacitors. Coincidentally, these framework conditions create precisely such an opportunity for construction of a sub-digital covert channel as we have shown before. The only difference in this case is that modulation is not performed via slight variations in rise time or signal phase, but in modulation of the radio frequency amplitude that is emitted via an antenna.

Consider an Radio Frequency Identification (RFID) reader that controls a door lock. Users get special RFID tokens which they can hold in front of the RFID reader. The reader performs a cryptographically secured handshake with the token and, upon successful verification that the token is legitimate, opens the door. Such RFID readers are manufactured by many different

companies and they all usually provide the functionality that a user can set and store their own cryptographic keys on the device. This is to ensure that the owner of the locking system is the only one who knows the keys and can issue new, genuine, tokens. We show that a vendor might have equipped such a product with a backdoor that allows gaining access to the system in an almost undetectable fashion by utilizing covert channels as described.

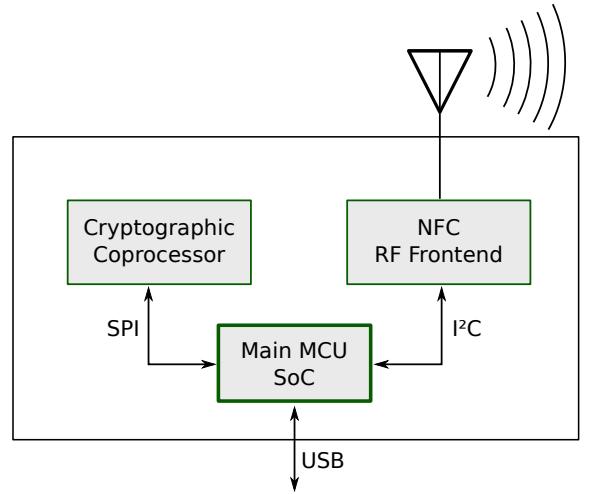


Fig. 10. Notional RFID reader with its components

Consider such a notional RFID reader and its basic design which is shown in Fig. 10. The board consists of a main processor (SoC) that is connected to various peripherals using different buses. One is a cryptographic coprocessor which contains master secrets that are written into the chip as part of the commissioning of the reader board. Communication

between the cryptographic coprocessor and SoC is encrypted and authenticated over a Serial Peripheral Interface (SPI). To communicate with near-field communication (NFC) tokens, a standard RF front end IC is used that is connected to the antenna. Successful authentication requests on the NFC interface are propagated over USB to an external lock control unit.

In such a system, multiple sub-digital covert channels could be existent: The most obvious one is a covert channel on the SPI. If the chip manufacturer embedded a sub-digital covert channel in the cryptographic coprocessor then the coprocessor itself could be sending out its master secret periodically on the SPI, for example by using rise time modulation as we described before. These secrets are sufficiently short (256 bit) such that transmission can be extremely slow (in the range of 1 bit per minute to prevent detection) and still would allow an attacker to perform relatively fast recovery of the secrets (in around 4,5 hours). From an attacker's point of view, this channel is hard to exploit because it requires either physical access or high-end radio equipment.

If the firmware of the SoC itself were to be compromised, then multiple covert channels could be constructed. Again, the most obvious one is an on-board channel over any of the three connected interfaces. The manufacturer of the RFID reader would be able to steal keys of units in the field that have been parameterized by customers, but again physical access to the hardware itself or sophisticated equipment would be necessary.

The most interesting channel, however, is the one we hinted at in the introduction of this section: Usually, RF front end ICs allow software tuning of antennas in order to compensate for small physical differences that occur during antenna manufacturing [22]. During production of the system, the antenna is connected to the RF front end IC and software calibration is performed in order to achieve impedance matching between both components and therefore maximize field strength. This means that it is possible to deliberately attenuate the RF signal in software by introducing slight impedance mismatches and effectively modulate the field in a user-customizable fashion. Such a channel could leak keys that are required for successful authentication over the RF interface. An attacker who would have implanted this channel into an RFID reader would need very little effort to perform channel data recovery.

An attacker who knows that the RFID reader is compromised could analyze the modulated RF stream externally using special receiving equipment (an oscilloscope would be sufficient for regular 13.56 MHz HF-RFID communication) and learn the keys that are required to authenticate against the reader, effectively creating a backdoor.

All these channels have in common that they are sub-digital in the sense that during test and auditing there will usually be equipment used that cannot display the presence of the covert channel because it is not part of the digital representation. Concretely, when using a logic analyzer to look at the transmitted bits there will be no difference visible between a system that is backdoored and one that is not, precisely because the channel itself is only present in the

gray area of encoding ambiguity of the analog interpretation of digital signals. Similarly, the covert channel in the RF case will not survive demodulation because the physical differences that constitute the covert signal are so small that the demodulator is indifferent towards them.

VI. CONCLUSION

We have demonstrated the existence and feasibility of intra-packet physical layer covert channels. We implemented such channels on commodity hardware by abusing already present anti-EMI facilities. Concretely, we transmitted data via the Spread Spectrum Clocking unit and also showed that the approach works by modulating the rise time of an arbitrary output pin. Thus, we have demonstrated the practical feasibility of such attacks and by evaluating the channel capacity on real hardware, we have shown that transmission bandwidth is still high enough to be a relevant threat because it could be used to leak cryptographic material.

These channels by definition exist on any device which allows control over electromagnetic emission countermeasure facilities. Since the speed of microcontrollers has increased significantly in the last years, the necessity for presence of such EMI countermeasure facilities has equally risen. Such countermeasure facilities are therefore already present in a wide variety of device by default as of today.

Something we wish to highlight is that data exfiltration is not merely limited to wire bound tapping like we showed in our examples. It would be practical for a real-world attacker with more sophisticated equipment (such as a high-end spectrum analyzer) to pick up the covertly transmitted data remotely. Our rationale why this is possible is as plausible as it is catchy: The only reason these EMI-countermeasure facilities are present in modern microcontrollers in the first place is because they cause an externally observable difference in radio frequency electromagnetic emission. It is the prerequisite for them to be effective. This means in turn that any EMI-manipulation regardless of its physical carrier will simultaneously cause subtle differences in the power levels within the EM spectrum. So even when wire bound protocols are affected primarily it is our estimate that remote, wireless data exfiltration is well within the arsenal available to a sophisticated attacker.

In future work, we wish to further explore the possibility of purely wireless data exfiltration. Another goal is to narrow the gap between the channel capacity we were able to achieve and the maximally achievable channel capacity by using more sophisticated equipment. This can be done by using special-purpose equipment such as an FPGA board in contrast to general-purpose equipment like an oscilloscope.

Furthermore, while we focused on wire-bound physical layer covert channels, wireless equivalents should warrant further research.

Finally, we have shown that for security auditing, it is insufficient to only examine the inter-packet timing characteristic and transmitted data on the data link layer. In order to detect physical layer covert channels, one has to dig deeper and take a look into the analog realm and *intra-packet* timing. The fact

that such channels can be constructed easily and cheaply with off-the-shelf hardware means that they are even simpler to incorporate in custom hardware.

Because of their asymmetry property they are invisible to standard receivers and therefore easy to miss. Since they pose a threat to confidentiality and system integrity, we hope that our work encourages further research in that area in order to reveal where such channels might already exist in today's real-world systems.

REFERENCES

- [1] J. Appelbaum, J. Horchert, and C. Stöcker. Shopping for spy gear: Catalog advertises NSA toolbox. *Spiegel Online International*, 2013. <http://www.spiegel.de/international/world/catalog-reveals-nsa-has-back-doors-for-numerous-devices-a-940994.html>.
- [2] J. Bauer, S. Schinzel, F. Freiling, and A. Dewald. Information leakage behind the curtain: Abusing anti-EMI features for covert communication. In *IEEE International Symposium on Hardware Oriented Security and Trust, HOST 2016, McLean, VA, USA, 3-5 May, 2016*.
- [3] R. Bracewell. *The Fourier Transform and Its Applications*. McGraw-Hill Science/Engineering/Math, 3 edition, June 1999.
- [4] S. Cabuk, C. E. Brodley, and C. Shields. IP covert timing channels: Design and detection. In *Proceedings of the 11th ACM conference on Computer and communications security*, pages 178–187, 2004.
- [5] C. J. Dietrich, C. Rossow, F. C. Freiling, H. Bos, M. van Steen, and N. Pohlmann. On botnets that use DNS for command and control. In *European Conference on Computer Network Defense (EC2ND)*, 2011.
- [6] M. M. Farag, L. W. Lerner, and C. D. Patterson. Interacting with hardware trojans over a network. In *HOST*, pages 69–74. IEEE, 2012.
- [7] D. Genkin, L. Pachmanov, I. Pipman, and E. Tromer. Stealing keys from pcs using a radio: Cheap electromagnetic attacks on windowed exponentiation. *Cryptology ePrint Archive, Report 2015/170*, 2015. <http://eprint.iacr.org/>.
- [8] S. Gianvecchio and H. Wang. Detecting covert timing channels: An entropy-based approach. In *Proceedings of the 14th ACM conference on Computer and communications security*, pages 307–316, 2007.
- [9] J. Giffin, R. Greenstadt, P. Litwack, and R. Tibbetts. Covert messaging through TCP timestamps. In *Workshop on Privacy Enhancing Technologies*, pages 194–208, 2002.
- [10] M. Gruhn and T. Müller. On the practicability of cold boot attacks. In *ARES*, pages 390–397. IEEE Computer Society, 2013.
- [11] J. A. Halderman, S. D. Schoen, N. Heninger, W. Clarkson, W. Paul, J. A. Calandriño, A. J. Feldman, J. Appelbaum, and E. W. Felten. Lest we remember: cold-boot attacks on encryption keys. *Commun. ACM*, 52(5):91–98, 2009.
- [12] T. Iakymchuk, M. Nikodem, and K. Kepa. Temperature-based covert channel in FPGA systems. In *ReCoSoC*, pages 1–7. IEEE, 2011.
- [13] L. Ji, W. Jiang, B. Dai, and X. Niu. A novel covert channel based on length of messages. In *International Symposium on Information Engineering and Electronic Commerce*, pages 551–554. IEEE, 2009.
- [14] R. A. Kemmerer. Shared resource matrix methodology: An approach to identifying storage and timing channels. *ACM Transactions on Computer Systems*, 1(3):256–277, Aug. 1983.
- [15] S. T. King, J. Tucek, A. Cozzie, C. Grier, W. Jiang, and Y. Zhou. Designing and implementing malicious hardware. In *Proceedings of the 1st USENIX Workshop on Large-scale Exploits and Emergent Threats*, pages 1–8, 2008.
- [16] B. W. Lampson. A note on the confinement problem. *Communications of the ACM*, 16(10):613–615, 1973.
- [17] Y. Liu, D. Ghosal, F. Armknecht, A.-R. Sadeghi, S. Schulz, and S. Katzenbeisser. Hide and seek in time - robust covert timing channels. In M. Backes and P. Ning, editors, *ESORICS*, volume 5789 of *Lecture Notes in Computer Science*, pages 120–135. Springer, 2009.
- [18] I. S. Moskowitz and M. H. Kang. Covert channels-here to stay? In *Computer Assurance, 1994. COMPASS'94 Safety, Reliability, Fault Tolerance, Concurrency and Real Time, Security. Proceedings of the Ninth Annual Conference on*, pages 235–243, 1994.
- [19] Motorola, Inc. *SPI Block Guide V03.06*, 2003.
- [20] S. Murdoch and S. Lewis. Embedding covert channels into TCP/IP. In *Information Hiding*, pages 247–261, 2005.
- [21] S. J. Murdoch. Covert channel vulnerabilities in anonymity systems. Technical Report UCAM-CL-TR-706, University of Cambridge, Computer Laboratory, Dec. 2007.
- [22] NXP Semiconductors. *MFRC522 Standard 3V MIFARE reader solution (112138) Rev 3.8*, September 2014.
- [23] NXP Semiconductors. *UM10204 I²C-bus specification and user manual*, April 2014.
- [24] Philips Semiconductors. *I²S bus specification*, 1997.
- [25] C. H. Rowland. Covert channels in the TCP/IP protocol suite. *First Monday*, 2(5), 1997.
- [26] S. Sadasivan. White paper: An introduction to the ARM Cortex-M3 processor. October 2006.
- [27] G. Shah and M. Blaze. Covert channels through external interference. In *Proceedings of the 3rd USENIX conference on Offensive technologies (WOOT09)*, pages 1–7, 2009.
- [28] ST Microelectronics. *RM0090 Reference manual STM32F405xx, STM32F407xx, STM32F415xx and STM32F417xx advanced ARM-based 32-bit MCUs*, September 2011.
- [29] ST Microelectronics. *ARM Cortex-M4 STM32F405xx STM32F407xx datasheet*, May 2012.
- [30] M. Tehranipoor and F. Koushanfar. A survey of hardware

- trojan taxonomy and detection. *IEEE Design & Test of Computers*, 27(1):10–25, 2010.
- [31] S. Wendzel and J. Keller. Systematic engineering of control protocols for covert channels. In *Communications and Multimedia Security*, pages 131–144, 2012.
- [32] A. X. Widmer and P. A. Franaszek. A DC-balanced, partitioned-block, 8B/10B transmission code. *IBM Journal of Research and Development*, 27(5):440–451, 1983.
- [33] S. Zander, G. Armitage, and P. Branch. A survey of covert channels and countermeasures in computer network protocols. *Communications Surveys & Tutorials, IEEE*, 9(3):44–57, 2007.