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.**

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 $f(t)$. To achieve resilience against additive noise $g(t)$, a digital interpretation function $\mathcal{D}$ is specified by the data link layer which translates noisy analog signals $f(t) + g(t)$ back into their original, unaltered digital representation.

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.

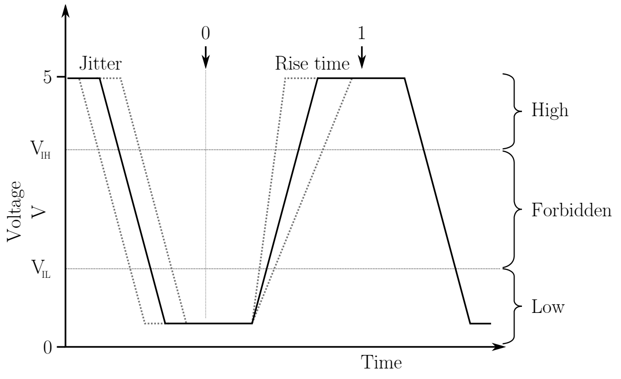

Fig. 1. Signal changes of jitter/rise time that preserve the physical property

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. 2. 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 [14] and the extraction of information from hardware by physical access [9, 10] 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

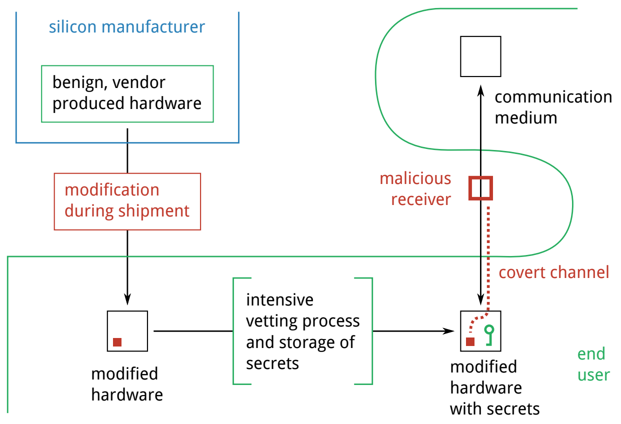
silicon manufacturer
benign, vendor produced hardware
modification during shipment
modified hardware
intensive vetting process and storage of secrets
communication medium
malicious receiver
covert channel
modified hardware with secrets
end user

Fig. 2. Example of supply chain poisoning

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 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 [15] 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 [13] 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 [8, 19, 23] and “in the wild” [1, 4]. 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 [12, 16, 20, 27]. 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 [3, 7, 17, 29]. 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 [5, 11, 14, 24, 26]. For example, work on hardware Trojans falls into this category, e.g., King et al. [14], Tehranipoor and Koushanfar [26], Farag, Lerner, and Patterson [5]. 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. [11] 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 [24] 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. [6]; 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, of which an extended version is available as a technical report [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.

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 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 I^2C, SPI, I^2S or USB [18, 21, 22].

II. Background: Anti-EMI Features

Digital signals which make fast transitions from one logic level into its inverse contain many high-frequency components. For system designers, this is an unwanted side effect because part of the energy contained within the signal will be radiated away as Electromagnetic Interference (EMI), in particular the high frequency components of the spectrum. If the slew rate of such signals were artificially limited, radiated emission would equally decrease. Microcontroller manufacturers are aware of this and equip many of the newer devices with the possibility to limit the rise and fall times of digital signals by use of special configuration registers [25]. According to the needs of the application, the analog properties of a digital signal can therefore be modified within certain limits. This fine-tuning is configurable in firmware during runtime and lays the foundation for our approach.

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 to construct our covert channel.

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 Vcc = 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. 3. It was captured using a Tektronix MSO4034.

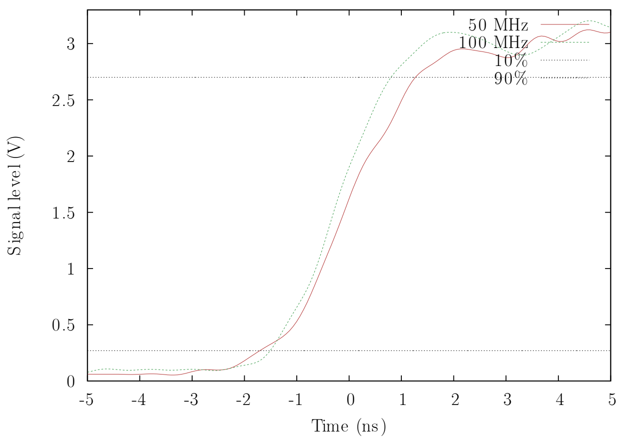

Fig. 3. Signal differences of 50 MHz and 100 MHz drivers

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, I^2C, SPI, etc.).

IV. Data Encoding

Depending on the number of discrete states that a receiver can discriminate, an appropriate encoding must be chosen for transmission of data. Concretely, we used binary encoding for the rise time approach of Sect. III. Note that the choice of encoding is independent of the type of channel itself.

When using binary encoding, the clock recovery of the demodulated covert channel is more complex than with an approach that uses more than two symbols. 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 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, whitening techniques could be used to keep signal disparity to a minimum [28].

An example of the actual measurement data that we did this type of clock recovery on can be seen in Fig. 4. In summary, with two symbols we achieved recovery speeds of about 5 Baud.

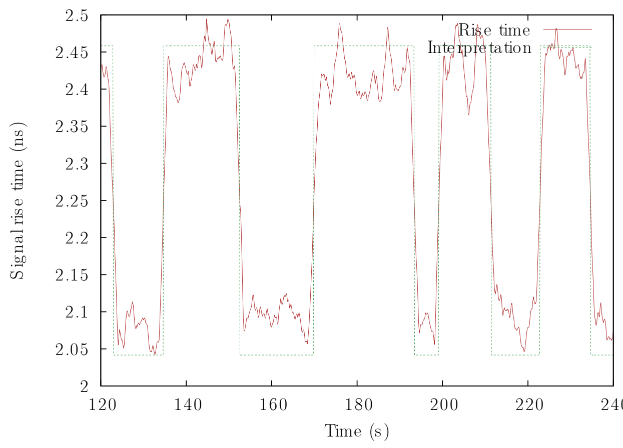

Fig. 4. Rise time over time and the conditioned signal

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 channel that we demonstrate experimentally in Sect. III modulates GPIO rise times for data transmission. We used a $b_o = 115.2\text{kBd}$ transport, but had only a covert symbol rate $b_c = 75.9\text{kBd}$ because we chose to overtly transmit plain text. The switching time t between states is almost negligible, in our case it was 120ns for each switch.

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

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

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 $B = 227.3\text{kBd}$. 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 receiver. Since we relied on general-purpose equipment, the recovery bottleneck was the USBTMC transmission of the results to the decoding PC.

V. 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. Even though the channel capacity is small, its 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.

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. Technical Report CS-2016-03, University of Erlangen, Dept. of Computer Science, Mar. 2016.

[3] 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.

[4] 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.

[5] M. M. Farag, L. W. Lerner, and C. D. Patterson. Interacting with hardware trojans over a network. In *HOST*, pages 69–74. IEEE, 2012.

[6] 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/.

[7] 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.

[8] J. Giffin, R. Greenstadt, P. Litwack, and R. Tibbetts. Covert messaging through TCP timestamps. In *Workshop on Privacy Enhancing Technologies*, pages 194–208, 2002.

[9] M. Gruhn and T. Müller. On the practicability of cold boot attacks. In *ARES*, pages 390–397. IEEE Computer Society, 2013.

[10] J. A. Halderman, S. D. Schoen, N. Heninger, W. Clarkson, W. Paul, J. A. Calandrino, 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.

[11] T. Iakymchuk, M. Nikodem, and K. Kepa. Temperature-based covert channel in FPGA systems. In *ReCoSoC*, pages 1–7. IEEE, 2011.

[12] 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.

[13] 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.

[14] 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.

[15] B. W. Lampson. A note on the confinement problem. *Communications of the ACM*, 16(10):613–615, 1973.

[16] 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.

[17] 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.

[18] Motorola, Inc. *SPI Block Guide V03.06*, 2003.

[19] S. Murdoch and S. Lewis. Embedding covert channels into TCP/IP. In *Information Hiding*, pages 247–261, 2005.

[20] S. J. Murdoch. Covert channel vulnerabilities in anonymity systems. Technical Report UCAM-CL-TR-706, University of Cambridge, Computer Laboratory, Dec. 2007.

[21] NXP Semiconductors. *UM10204* I^2C*-bus specification and user manual*, April 2014.

[22] Philips Semiconductors. I^2S *bus specification*, 1997.

[23] C. H. Rowland. Covert channels in the TCP/IP protocol suite. *First Monday*, 2(5), 1997.

[24] 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.

[25] ST Microelectronics. *RM0090 Reference manual STM32F405xx, STM32F407xx, STM32F415xx and STM32F417xx advanced ARM-based 32-bit MCUs*, September 2011.

[26] M. Tehranipoor and F. Koushanfar. A survey of hardware trojan taxonomy and detection. *IEEE Design & Test of Computers*, 27(1):10–25, 2010.

[27] S. Wendzel and J. Keller. Systematic engineering of control protocols for covert channels. In *Communications and Multimedia Security*, pages 131–144, 2012.

[28] 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.

[29] 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.