Leakier Wires: Exploiting FPGA Long Wires for Covert- and Side-Channel Attacks

Ilias Giechaskiel, Ken Eguro, Kasper B. Rasmussen

ACM Transactions on Reconfigurable Technology and Systems,
vol. 12, no. 3, pp. 11:1–11:29, August 2019

Leakier Wires: Exploiting FPGA Long Wires for Covert- and Side-Channel Attacks

ILIAS GIECHASKIEL, University of Oxford, United Kingdom
KEN EGURO, Microsoft, United States of America
KASPER B. RASMUSSEN, University of Oxford, United Kingdom

In complex FPGA designs, implementations of algorithms and protocols from third-party sources are common. However, the monolithic nature of FPGAs means that all sub-circuits share common on-chip infrastructure, such as routing resources. This presents an attack vector for all FPGAs that contain designs from multiple vendors, especially for FPGAs used in multi-tenant cloud environments, or integrated into multi-core processors. In this paper, we show that "long" routing wires present a new source of information leakage on FPGAs, by influencing the delay of adjacent long wires. We show that the effect is measurable for both static and dynamic signals, and that it can be detected using small on-board circuits. We characterize the channel in detail and show that it is measurable even when multiple competing circuits (including multiple long-wire transmitters) are present and can be replicated on different generations and families of Xilinx devices (Virtex 5, Virtex 6, Artix 7, and Spartan 7). We exploit the leakage to create a convert channel with 6 kbps of bandwidth and 99.9% accuracy, and a side channel which can recover signals kept constant for only 1.3 µs, with an accuracy of more than 98.4%. Finally, we propose countermeasures to reduce the impact of this information leakage.[1]

CCS Concepts: • **Security and privacy → Side-channel analysis and countermeasures**; *Embedded systems security*; • **Hardware → Reconfigurable logic and FPGAs**.

Additional Key Words and Phrases: FPGA covert channel; information leakage; long wire delay; crosstalk

ACM Reference Format:
Ilias Giechaskiel, Ken Eguro, and Kasper B. Rasmussen. 2019. Leakier Wires: Exploiting FPGA Long Wires for Covert- and Side-Channel Attacks. *ACM Trans. Reconfig. Technol. Syst.* 12, 3, Article 11 (August 2019), 29 pages. https://doi.org/10.1145/3322483

1 INTRODUCTION

The ever-increasing size and sophistication of Field-Programmable Gate Arrays (FPGAs) make them an ideal platform for System-on-Chip (SoC) integration. FPGAs are often used in high-bandwidth, low-latency applications, providing functionality such as network card replacement, or massively parallel computation. Besides permeating distributed systems and critical infrastructure, FPGA chips are also integrated in end-products, ranging from consumer electronics to medical and scientific equipment. As a result, protecting their security is necessary to ensure that their computations are performed in a trustworthy manner.

[1]This article extends a paper presented at ASIACCS 2018 [10].

Authors' addresses: Ilias Giechaskiel, University of Oxford, Oxford, United Kingdom, ilias.giechaskiel@cs.ox.ac.uk; Ken Eguro, Microsoft, Redmond, Washington, United States of America, eguro@microsoft.com; Kasper B. Rasmussen, University of Oxford, Oxford, United Kingdom, kasper.rasmussen@cs.ox.ac.uk.

1936-7406/2019/08-ART11 $15.00
https://doi.org/10.1145/3322483

The high cost of design and development has led to an increase in outsourcing, making it common to have designs from different contractors on the same FPGA chip. Such designs often include protocol and data structure implementations, or more sophisticated circuits, like radio front-ends or soft processors. This practice raises concerns about the malicious inclusion of circuits (cores) that have additional backdoor functionality, especially when FPGAs are integrated in multi-tenant cloud environments, or multi-core processors. Although third-party implementations can be functionally validated before being included in the overall design, it is not always possible to detect unintentional information leakage or intentional covert channels [12]. It is therefore important to identify sources of information leakage and protect against any resulting channels exploiting those sources.

In this paper, we show that the value driven onto certain types of FPGA routing resources, called "long" wires, influences the delay of nearby wires, *even when the driven value remains constant.* This distinguishes our approach from prior work which depends on fast-changing signals [9, 15, 40], and thus local voltage drops or inductive crosstalk. Specifically, we find that if a long wire carries a logical 1, the delay of nearby long lines will be slightly lower than when it carries a logical 0. This difference in delay allows cores sharing the same reconfigurable FPGA fabric to communicate, even when they are not directly connected.

We demonstrate the phenomenon by building a transmitter and receiver, which are unconnected, and only use adjacent long wires to communicate. The receiver is a three-stage Ring Oscillator (RO), whose routing uses a long wire between two of its stages. The transmitter drives a long wire adjacent to that of the RO. When the transmitting wire carries a logical 1, the routing delay of the RO long wire decreases, thereby increasing the RO frequency. We detect these minor frequency changes by counting the number of the RO signal transitions during a fixed time interval. This mechanism can be used for covert communication and for the exfiltration of fast, dynamic signals.

We conduct extensive experiments on four Xilinx FPGA families and show that the phenomenon is independent of the device used, the location and orientation of the transmitter and receiver, and the pattern of transmission. We perform all tests on stock prototyping boards without modifications, and show that the phenomenon can be detected even in the presence of environmental noise and with only small circuits internal to the FPGA. Using this information leakage, we are able to create a convert channel with bandwidth upwards of 6 kbps and 99.9% accuracy, as well as a side channel which can recover signals which are kept constant for as low as 128 cycles (1.3 µs), with an accuracy of more than 98.4%. Finally, we propose new defense mechanisms which can be implemented by systems and tools designers to reduce the impact of this information leakage.

2 BACKGROUND

FPGAs are integrated circuits that implement reconfigurable hardware. At a basic level, they consist of blocks of configurable lookup tables (LUTs), which can be used to represent the truth table of combinatorial functions. They also include registers to store data, as well as programmable routing, which determines how the LUTs and registers are interconnected. FPGAs can thus be used to represent all computable functions, including emulating sophisticated circuits such as entire CPUs.

The Xilinx FPGAs used in our experiments internally have a grid layout, whose fundamental building block is called a *Configurable Logic Block* (CLB). It is composed of two *slices*, each of which contains four LUTs and registers. Each CLB has an associated *switch matrix*, which contains resources to connect elements within a CLB, and enables CLBs to communicate with each other. There are multiple types of such communication wires, which have different orientations and lengths. In this paper we focus on a specific type of routing resource, called a *long* wire. Long wires are used to efficiently communicate between CLBs that are far apart, and can be *vertical* (connecting elements with the same x coordinate), or *horizontal* (same y coordinate). We have observed the phenomenon in both types of wires, but for brevity we limit our discussion to vertical longs, or

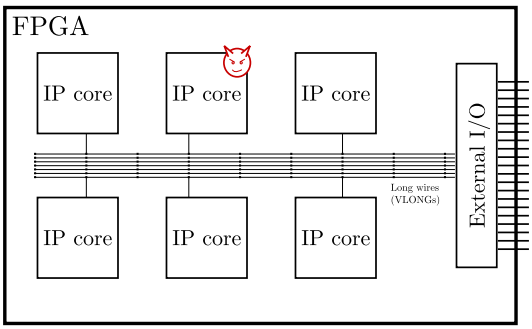


Fig. 1. System model. Different IP cores share common FPGA routing resources. The cores can be supplied by different contractors and may contain malicious functionality.

VLONGs. Due to the FPGA's routing topology, additional shorter wires are often needed to connect certain elements via long wires. We will refer to these wires as "local routing".

Usually, the details of how logic elements are placed and signals are routed are hidden from the circuit designers. Designers define their desired logic, but the conversion to a physical implementation is handled by the manufacturer's tools. Compiler directives for the manual routing of signals are available, but these are rarely used. In the absence of manual directives, the tools may elect to use any wire, including longs, to carry a given signal in the circuit, without alerting designers.

That said, user-designed circuits often share the FPGA with third-party implementations of various protocols, data structures, and algorithms. These licensed designs, called Intellectual Property (IP) cores or blocks, often come in a pre-routed black-box format, to eliminate the variability of on-the-fly routing and attain a known clock frequency. As a result, the routing of these blocks is opaque to circuit designers, and blocks created by different parties can use routing resources in the same channel of long wires. As our paper shows, this use of nearby long wires can enable malicious circuits to communicate covertly, or extract information from other cores.

Ring Oscillators (ROs) are a type of circuit which consists of an odd number of NOT gates, chained together in a ring formation (i.e., the output of the last gate is fed back as input to the first gate). ROs form a bi-stable loop, whose output oscillates between 1 and 0 (true and false). The frequency of oscillation depends on the number of stages in the RO, the delay between the stages, as well as voltage, temperature, and small variations in the manufacturing process [11]. ROs in FPGAs are used as temperature monitors [42], True Random Number Generators (TRNGs) [38], and Physical Unclonable Functions (PUFs) [23], while in this paper we present a way to use them to detect the logic state of nearby long wires.

3 SYSTEM AND ADVERSARY MODEL

FPGA designs contain IP cores sourced from third-parties, and some of these cores may contain unwanted functionality, as shown in Figure 1. These third-party IP cores can be distributed as fully-specified, pre-placed, and pre-routed elements ("macros") to meet timing constraints (e.g., DDR controllers) and reduce compilation time, with the macro repositioned at specific intervals where the logic and routing fabric is self-similar [16, 18–20].

As FPGAs often process highly-sensitive information (e.g., cryptographic keys), it is essential to ensure that data does not leak to unauthorized third-parties. In this paper, we focus on malicious IP cores which aim to infer information about the state of nearby (but physically unconnected) logic. The adversary can thus insert one or more IP cores into the design, but these cores are not

directly connected. The adversary can also define the internal placement and routing of his/her own blocks and force these cores to use specific routing resources that can compromise the integrity of a reverse-engineered target IP block. Note that directly connecting to the target IP block would result in a logical error in the compilation flow, but merely using adjacent wires does not raise such errors. We discuss how adversaries can accomplish their goals in Section 3.2.

The adversary does not have physical access to the board, and can thus not alter the environmental conditions or physically modify the FPGA board in any way. There is also no temperature control beyond the standard heatsink and fan already mounted on the FPGA (if any), and we do not add any special voltage regulation, or shielding to the chip or the connected wires. Such modifications reduce noise and improve the stability of measurements [21, 26, 35, 42], and would thus make the adversary's attacks easier.

In this paper, we show that by using long wires, an adversary can infer the nearby state of blocks he/she does not control (eavesdrop), or establish covert communication between two co-operating IP cores under his/her control, even in the presence of power and temperature fluctuations. Our threat model thus covers attacks on FPGAs used in the cloud, as well as FPGAs integrated into multi-core CPUs. We provide further motivation and applications of the capabilities offered by this new source of information leakage in Section 3.1.

3.1 Motivation

With increased outsourcing, Hardware Trojans (HTs) have become a common-place security threat for FPGAs [7, 36]. Adversarial IP cores can thus eavesdrop on nearby cores and attempt to extract information about their state. As designs are often tested to detect HTs and other security threats [15, 21, 40], we assume that IP cores provide legitimate functionality that is needed by the user, and that they do not contain additional logic which would make them easy to detect. Indeed, the transmitter and receiver we present have dual use, hiding their malicious functionality in their routing, not their actual combinatorial and sequential logic. As a result, unlike conventional backdoors, our IP cores would pass timing/netlist/bitfile verification, since they do not require additional gates, presenting a bigger challenge to designers.

Multi-user setups present further threats beyond a malicious core eavesdropping on signals not under the adversary's control. Intel Xeon and other CPUs with integrated FPGAs bring FPGAs closer to a traditional server model, while FPGAs in cloud environments (e.g., Amazon EC2 F1 instances) are also becoming increasingly available. Although these are currently allocated on a per-user basis, we can expect that they will eventually become sharable commodity resources, since FPGAs already allow for partial reconfiguration, and designs exist where different processors have access to and can re-configure the same FPGA chip [33]. Indeed, this threat model is becoming commonplace when considering the security of FPGA designs in future applications [10, 27, 41].

An additional threat arises when IP cores of different security guarantees are integrated on the same design [12, 13, 32]. For example, an adversary implementing the FM radio core on a phone SoC would want to eavesdrop on the Trusted Platform Module's (TPM) AES encryption operations to recover its key. As sensitive cores are highly scrutinized, an adversary who has also implemented the TPM would want to establish a covert channel to transfer the key using an inconspicuous transmitter. Finally, the same phenomenon can be exploited to watermark circuits [5, 31], or introduce a no-contact debugging mechanism to detect stuck signals, without altering routing.

3.2 Influencing Placement and Routing

A potential issue with pre-placed and pre-routed IP cores is that they are specific to an FPGA generation (but can be used in different devices within the same family). As we show in Section 7.2, however, the phenomenon we present persists across four generations of Xilinx chips. As a result,

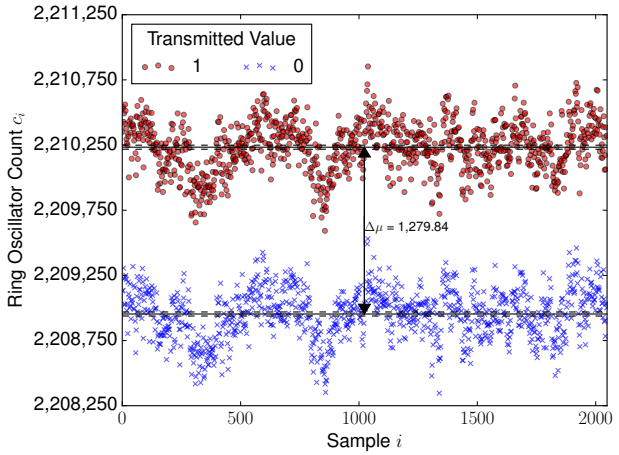


Fig. 2. Ring oscillator counts and 99% confidence intervals for a transmitter and receiver using 5 longs each. The receiver is able to distinguish between signals using a simple threshold, despite environmental noise.

an adversary can provide an IP generation wizard that provides different routing for different families, and dynamically choose the placement of the IP cores. In fact, as we show in Section 8, the location of the actual logic and wires is not important, so the adversary merely needs to ensure that the transmitter and the receiver use long wires which are adjacent.

An adversary who only pre-routes but does not pre-place cores can still succeed, even if the absolute placement of his/her cores is left to the routing tools. Assume the FPGA has N long wires, the transmitted signal can be recovered from w nearby wires, the receiver uses R longs, and the transmitter uses T longs. Then, the probability that at least one segment of the transmitter is adjacent to a segment of the receiver is $(R + T - 1) \cdot w/N$, assuming the tools place the two cores at random. For the FPGA boards we have used, $N \approx 8,500$ (equal to the number of CLBs) and $w = 4$, so with $R = T = 5$, an adversary has a 0.42% chance of success. Since tools do not pick locations at random or spread the logic, the probability of success is higher in practice. The adversary can also increase this probability by accessing relatively unique elements such as Block RAM (BRAM), DSP blocks, or embedded processors on the FPGA fabric. For example, the devices we used have less than 150 DSP slices and 300 BRAM blocks, so accessing them reduces the number of possible placements for the attacker's cores.

A more powerful adversary can instead subvert the compilation tools themselves, which is a common threat model for FPGAs [12, 17]. Note that, as before, since the final netlist itself is often verified post-synthesis and -routing, the adversary still does not desire to include additional logic in the design, but just affect the routing/placement of his/her malicious cores. Finally, in co-located multi-user instances, the adversary *is* the user, so he/she can always choose the location of his/her own cores, without the need to rely on the above.

4 CHANNEL OVERVIEW

Our channel exploits the fact that the delay of long wires depends on the logical state of nearby wires, *even when the signals they are carrying are static*. We find that when the transmitting wire carries a 1, the delay of the nearby receiving wire is lower, which results in higher ring oscillator (RO) counts. This is a distinct mechanism from prior research, which depends on the switching activity

of nearby circuits, and which instead decreases RO frequency [15, 40], as we also independently verify in Section 6.3.

This dependence on the logical state of the transmitter is exemplified in Figure 2. The red dots and blue x's are RO counts when the transmitter wire carries a logical 1 or 0 respectively. The difference between the counts when transmitting 1s and 0s is clear, even under local fluctuations due to environmental and other conditions: even when the absolute frequencies of the ring oscillator change, the difference between the two frequencies remains the same.

In order to characterize the efficacy and quality of the communication channel in detail, we perform a number of experiments, the setup of which is detailed in Section 5. We first show in Section 6 that the strength of the effect does not depend on the transmission pattern, by measuring the effect of an alternating sequence of 0s and 1s, as well as that of long runs of 0s and 1s, and of pseudo-random bits. We illustrate that even for fast-changing dynamic signals, an eavesdropping attacker can obtain the fraction of 1s and 0s, i.e., the Hamming Weight on the transmitting wire.

We then show that longer measurement periods and overlaps make it easier to distinguish between different bits in Section 7. The strength of the effect changes based on the receiver and transmitter lengths, and this dependence exists across at least four generations of devices, but with a different magnitude. We also demonstrate that the absolute location and orientation of the transmitter and receiver do not change the magnitude of the effect in Section 8.

In Section 9 we show that the channel remains strong, even if significant computation is happening elsewhere on the device simultaneously, demonstrating that the channel can be used in a realistic environment. We demonstrate that for the transmitted information to be detectable, the transmitter and receiver wires need to be adjacent, but where exactly and in what direction the overlap occurs is not significant. We show that using two adjacent transmitters, the strength of the phenomenon increases in the same way as it does with longer wire overlaps (Section 10). This allows an adversary to increase bandwidth and reduce errors in the covert channel case, or bypass local balancing security measures in the side-channel case. These facts indicate that it may be difficult for designers to protect themselves from eavesdropping, or detect malicious transmissions.

Overall, we show that the channel is stable across FPGA generations, devices, and locations within a device. It can also be used to implement high-bandwidth covert communications and eavesdropping attacks, without tapping into existing signals, and with minimal resources (Section 11).

5 EXPERIMENTAL SETUP

In order to test the properties identified in Section 4, we need to determine the factors we wish to vary, keeping the rest of the setup fixed. This distinction naturally divides our experimental setup into two parts, as shown in Figure 3. The communication channel circuit contains just the transmitter and the ring oscillator receiver. The measurement half works independently of any specific channel implementation, generating the transmitted signal, sampling the RO counter, and transferring the data to a PC for analysis.

The bulk of our experiments are conducted on three Virtex 5 XUPV5-LX110T (ML509) evaluation boards. The boards include a heatsink and a fan, but we do not otherwise control for temperature, and we also do not modify the board in any way (e.g., by bypassing the voltage regulator) in accordance with our threat model. Each experiment is run on every device 5 times, collecting 2048 data points per run, and results are reported at the 99% confidence level.

5.1 Transmitter and Receiver

To illustrate the information leakage, our setup employs a minimal *transmitting* circuit: the transmitter consists of a buffer LUT that drives one or more long-wire segments connected end-to-end. We use the term transmitter for brevity and because in the controlled experiments we choose the

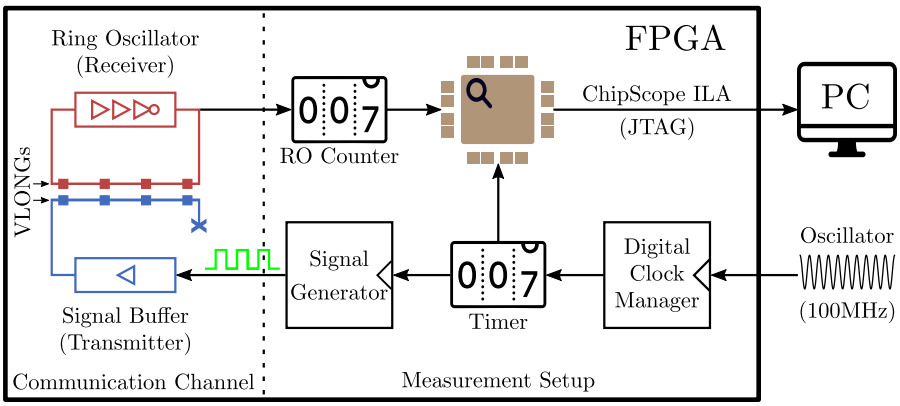


Fig. 3. Experimental setup. The transmitter and the receiver use adjacent long wires to communicate, but are otherwise unconnected. The measurement setup updates the signal to the transmitter every 2^n clock cycles, and measures the signal's effect on the ring oscillator frequency. The ring oscillator counts are transferred to the PC using ChipScope's ILA core for further analysis and processing.

value on the long wire, but the conclusions we draw are valid whether transmissions are intentional or not. In other words, our results apply to both covert- and side-channel communications (also see Section 11). The *receiving* circuit also uses long wires that are adjacent to the transmitter's wire segments. To measure the delay of the receiver's long wire segment(s), we include it as part of a three-stage ring oscillator. As in the experiments by Sun et al. [35], the oscillator contains one inverter (NOT gate) and two buffer stages, although we have also verified that one can equally use different structures (e.g., three NOT gates, or more stages). The wire's delay directly influences the frequency of oscillation, which we estimate by feeding the output of one of the RO stages to a counter in our measurement setup.

The receiver and the transmitter are initially on fixed locations of the device, but we change the location in Section 8 to show that it does not influence our measurements. We also change their lengths in Section 7.2 to show that the effect becomes more pronounced the longer the overlap is.

5.2 Measurement Setup

The measurement component generates the signals to be transmitted and measures the RO frequency. A new trigger event is produced every $N = 2^n$ clock ticks. At every trigger, the RO counter is read and reset, and a new value is presented to the transmitter. For most experiments, the signal generator simply alternates between 0s and 1s, but we change the pattern in Section 6 to show the generality of the channel.

The 100MHz system clock is driven by a Digital Clock Manager (DCM) to ensure clock quality. For the majority of our experiments, we fix $n = 21$ (corresponding to 2^{21} clock ticks, or 21 ms), but vary n in Section 7.1 to explore the accuracy vs. time trade-offs. The sampled data is transferred to a PC for analysis through Xilinx's ChipScope Integrated Logic Analyzer (ILA) core.

Unlike the circuit described above, the measurement logic is not hand-placed or hand-routed, due to the large number of experiments performed. Although the measurement logic could influence the RO frequency [25], we repeat our experiments on multiple locations, control for other patterns, and average over relatively lengthy periods of time. Moreover, in some experiments, we forgo using the ILA core entirely, and instead control the experiments using the Universal Asynchronous Receiver/Transmitter (UART) to ensure that the phenomenon is not caused by the core itself. Thus,

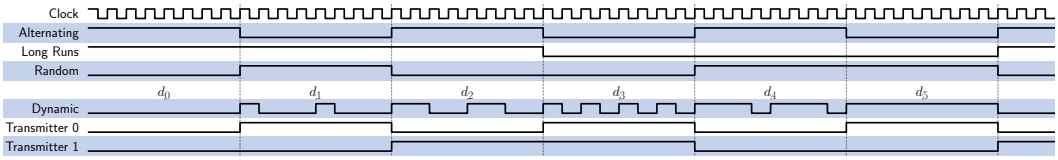


Fig. 4. Timing diagram for the various transmission patterns used in the experiments. We test patterns which remain constant within a measurement period (*Alternating, Long Runs, Random*), fast-changing patterns (*Dynamic*), and multiple simultaneous transmissions (*Transmitter 0* and *1*).

we believe that any effects of the measurement circuitry would influence the transmission of both zeros and ones equally, a hypothesis we further confirm in Section 9 by observing that the channel is only affected by adjacent wires.

5.3 Relative Count Difference

When a clock of frequency f_{CLK} is sampled every m ticks and a ring oscillator of frequency f_{RO} driving a counter measures c ticks, then $f_{RO}/f_{CLK} \approx c/m$, with an appropriate quantization error due to the unsynchronized nature of the RO and the system clock. Thus,

$$\frac{f_{RO}^1 - f_{RO}^0}{f_{RO}^1} \approx \frac{C^1 - C^0}{C^1} \tag{1}$$

where C^i and f^i represent the count and respective frequency when the transmitter has value i. As a result, the relative change of frequency can be approximated by using just the measured counts, irrespective of the measurement and clock periods.

In the basic setup, the transmitter alternates between sending zeros and ones. We denote the i-th sampled count as c_i, so the pair $p_i = (c_i, c_{i-1})$ always corresponds to different transmitted values. For the sake of notation clarity, we will assume that c_{2i+1} corresponds to a transmitted 1 and we will be using the quantity

$$\Delta RC_i = \frac{c_{2i+1} - c_{2i}}{c_{2i+1}}$$

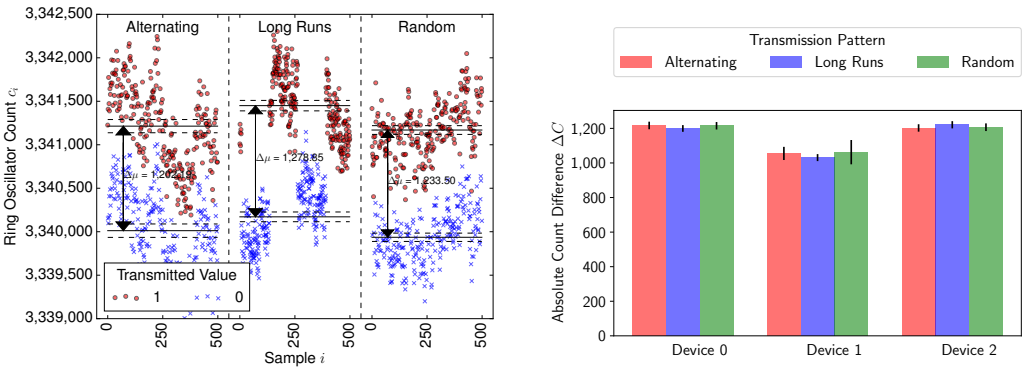
to indicate the relative frequency change between a transmitted one and zero. ΔRC will denote the average of ΔRC_i over all measurement pairs i. We discuss different transmission patterns in Section 6 and how to exploit the measurements in Section 11.

6 TRANSMITTER PATTERNS

In this section we show that the phenomenon observed does not depend on the pattern of transmissions, i.e., that only the values carried by the wire during the period of measurement matter, and not the values that precede or follow it. We first show this for relatively constant signals (Section 6.1), and then for highly dynamic ones (Section 6.2). Finally, we compare our results to those produced by switching activity, which is traditionally discussed in the context of Hardware Trojan detection (Section 6.3), and delay discussion of simultaneous transmissions until Section 10.

6.1 Constant Signals

In the default setup, we use a slowly alternating signal, where the transmitted value changes every sampling period. This pattern is denoted by *Alternating* in Figure 4. In this experiment, the transmitted value still remains constant within a given measurement period, and we sample the ring oscillator at the same default rate (every 21 ms), but change how the signal generator chooses the next value to be transmitted. The first additional pattern we test greatly slows down



(a) Example run

(b) Comparison across devices

Fig. 5. Effect of different static transmission patterns: (a) is a visualization of three different patterns: *Alternating* (left), *Long Runs* (middle), and *Random* (right). (b) is a comparison across devices, with 99% confidence intervals. The magnitude of the effect does not depend on the pattern used.

the alternation speed of the transmitted signal. This *Long Runs* pattern maintains the same value for 128 consecutive triggers—in essence, testing the effects of long sequences of zeros and ones. The second setup employs a Linear Feedback Shift Register (LFSR), which produces a pseudo-random pattern of zeros and ones, and is denoted by *Random* in Figure 4.

The results of this test are shown in Figure 5, with a sample of the data in Figure 5a, and a comparison across devices in Figure 5b. The RO counts remain significantly higher when transmitting a 1 versus a 0, and the average count difference remains identical, with almost no variability among the patterns. We deduce that the pattern of transmission has no persistent effect on the delay of nearby wires, allowing the channel to be used without having to ensure a balanced distribution of transmitted values – a property which is necessary when exploiting the information leakage to eavesdrop on nearby signals.

6.2 Dynamic Patterns

To show that the dominating factor in the observed phenomenon is the duration for which the transmitter remains at a logical 1, and *not* the switching activity of the circuit, we try various dynamic patterns. As a result, even if a signal is not sufficiently long-lived, the attacker can still deduce the signal's Hamming Weight (HW), and thus eavesdrop on signals he/she does not control. We explain in Section 11.2 how to use this property to recover secret state such as cryptographic keys through repeated measurements.

The dynamic patterns used are denoted by *Dynamic* in the timing diagram of Figure 4. During each sampling period, we loop the transmitter quickly through a 4-bit pattern at 100MHz. We test six different 4-bit patterns, only updating the looped pattern at each new sampling period. For example, for the pattern 1100 (d_2 in Figure 4), the transmitter would stay high for two 100 MHz clock ticks, then low for two clock ticks, then back to high for 2 ticks, etc., until the end of the sampling period. The six 4-bit patterns used are: $d_0 = 0000$, $d_1 = 1000$, $d_2 = 1100$, $d_3 = 1010$, $d_4 = 1110$, and $d_5 = 1111$. These patterns respectively have a HW of 0, 25, 50, 50, 75, and 100%, while their switching frequencies are 0, $f = f_{CLK}/8$, f, $2f$, f, and 0 respectively.

Figure 6 shows the average count C_i of the ring oscillator for each of pattern d_i. We see that the RO frequency increases with the Hamming Weight, so that $C_0 < C_1 < C_2 \approx C_3 < C_4 < C_5$. However, the

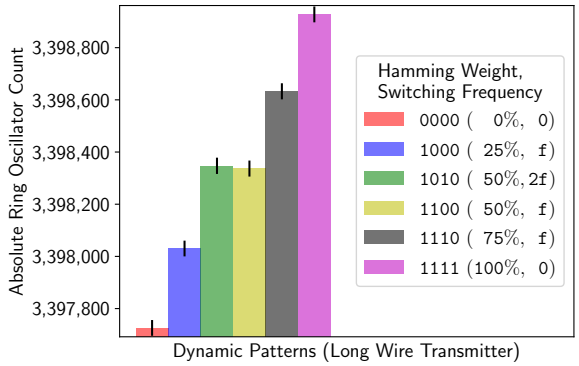


Fig. 6. Effect of dynamic switching activity using a long-wire transmitter. RO counts increase with the Hamming Weight, but not with the switching frequency.

frequency is otherwise unaffected by the switching transmission activity: the Kolmogorov-Smirnov test suggests that there is no statistically significant difference between the two distributions for d_2 and d_3. Note that the receiver would not able to distinguish between patterns d_2 and d_3 as a result (or more generally, any patterns with the same Hamming Weight), but we explain how to overcome this limitation in Section 11.2.

6.3 Local Routing

In this section, we show that when the two circuits do not have overlapping long wires, switching activity *decreases* the oscillation frequency of the RO. This reproduces the results reported by prior research on Hardware Trojan detection [15, 40] and also allows us to sanity-check our measurement setup. To test this dependence on the long wire overlap, we remove the transmitter using long wires, and replace it with a buffer of 312 consecutive LUTs packed into 39 CLBs, using only local intra- and inter-CLB routing. We then drive the same 6 dynamic patterns from Section 6.2 through the buffer, and summarize the measurements in Figure 7. We can clearly see that the ordering of the patterns exactly mirrors their relative switching activity, with the RO counts C_i corresponding to d_i decreasing with increased switching activity: $C_3 < C_1 \approx C_2 \approx C_4 < C_0 < C_5$. The difference between the patterns with the same switching activity d_1, d_2, d_4 is not significant according to the Kolmogorov-Smirnov test, but the count is slightly higher for d_5 compared to d_0, which have no switching activity. This suggests that the phenomenon we have identified and which reduces delay may be present for shorter wires as well, but is considerably weaker, and requires much bigger circuits. Overall, we can conclude that when the transmitter does not use longs which overlap with the receiver and generates a lot of switching activity through multiple redundant buffers, then the observed RO frequency is indeed reduced, reproducing the results of prior work.

7 MEASUREMENT PARAMETERS

In this section, we discuss the trade-offs between the quality of the channel and the measurement time (Section 7.1), and length of overlap between receiver and transmitter (Section 7.2).

7.1 Measurement Time

In this experiment, we return to the alternating pattern shown in Figure 4, and vary the measurement time by repeatedly quadrupling it. Both the absolute and the relative count difference for the various

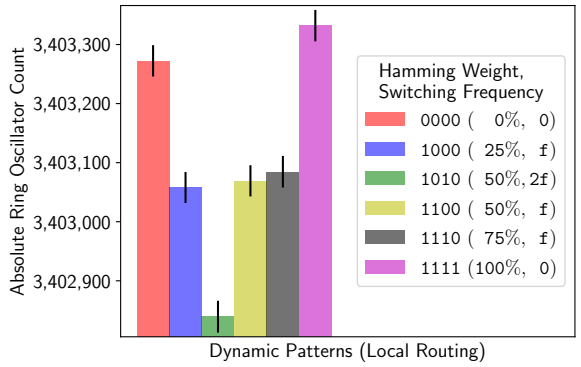


Fig. 7. Effect of dynamic switching activity without long-wire overlaps. RO counts decrease with switching frequency, and are almost unaffected by the Hamming Weight.

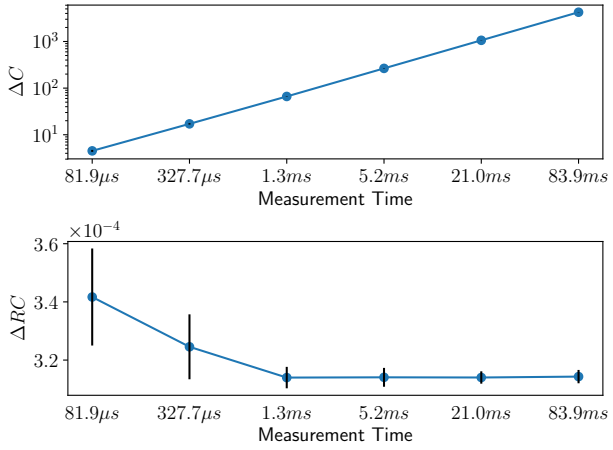
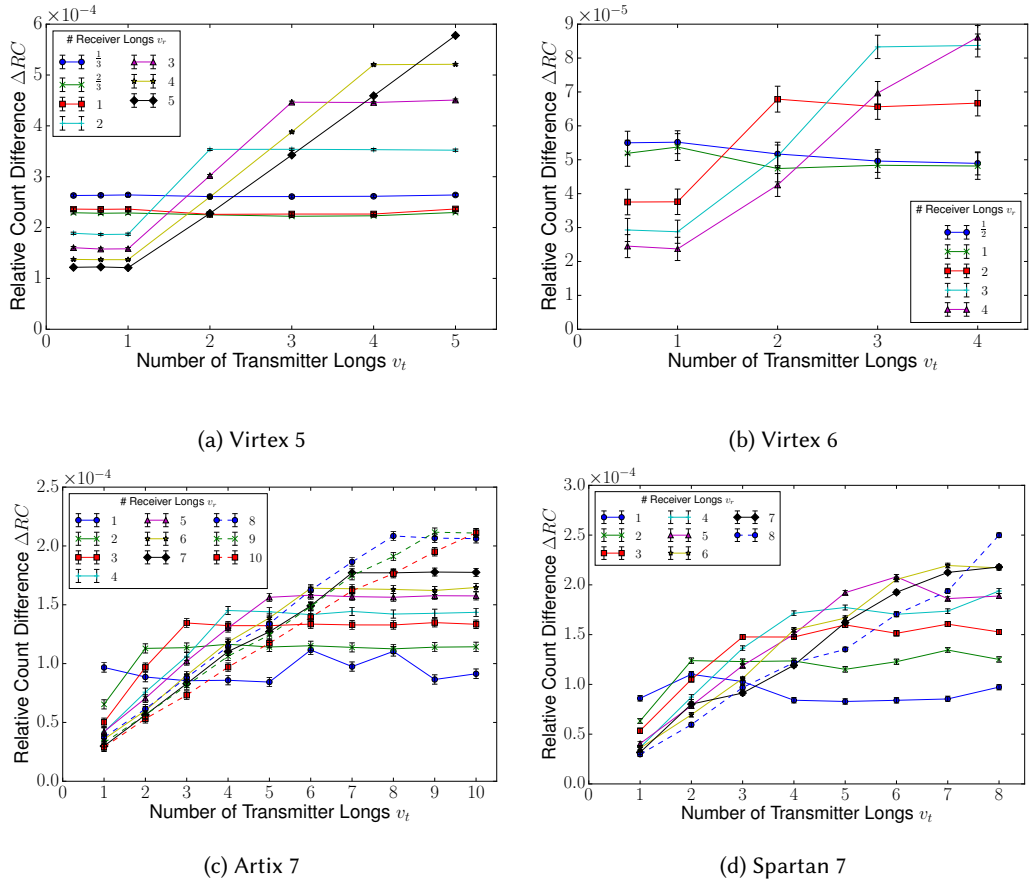


Fig. 8. Absolute and relative count differences and 99% confidence intervals for various measurement times. For a given transmitter and receiver overlap, the absolute magnitude of the effect as measured by the RO count difference increases linearly with time. For sufficiently long measurement periods, the relative count difference remains constant, but short measurement periods increase quantization errors and uncertainty.

times are shown in Figure 8. In the top of the figure, we see that the absolute count difference ΔC grows linearly with increasing measurement time. Hence, the RO count differences can be amplified proportionally to the duration of the measurement.

The relative differences ΔRC (shown in the bottom of Figure 8) remain approximately constant for measurement periods above 1 ms, in accordance with our theoretical prediction of Equation (1). The values for shorter measurement periods are still close, but are far noisier: for short measurement periods, the absolute difference is small ($\approx$4 for $t = 82\,\mu s$), increasing quantization errors, and making it harder to distinguish between signal and noise. These results indicate that for a given receiver/transmitter placement, the absolute magnitude of the effect depends solely on measurement time, with longer measurement periods making it easier to distinguish between signals and noise. An adversary can thus choose the measurement time, trading throughput for lower bit error rate.



(a) Virtex 5

(b) Virtex 6

(c) Artix 7

(d) Spartan 7

Fig. 9. Relative frequency changes ΔRC with 99% confidence intervals as a function of the transmitter and receiver lengths for different FPGA generations. The count difference is proportional to the overlap between the transmitter and the receiver.

7.2 Wire Length

We characterize the effect of varying the length (number) of transmitter and receiver wires v_t and v_r in four generations of devices. Besides the Virtex 5 devices we have been using so far, we also measure the effect on Virtex 6 ML605s, Artix 7 Nexys 4 DDRs and Basys 3s, and a Spartan 7 ArtyS7. The relative change in frequency ΔRC is shown for different combinations of v_t and v_r in Figure 9, for one device per generation. We notice the same common pattern for all devices.

For a given number of long wires v_r used by the ring oscillator, there are 3 distinct segments for ΔRC as the number of transmitter longs v_t increases. The first segment occurs for transmitters which use only parts of a long. Using partial wires is possible because even though `VLONG`s can only be driven from the top or the bottom, they have additional intermediate "taps" which can be used to read the values of the signal they carry. In practice, using partial wires does not have an effect on the strength of the phenomenon: ΔRC remains constant for all fractions of a long. This result is to be expected since, electrically, the entire long wire is driven even if the output tap does not take full advantage of its length.

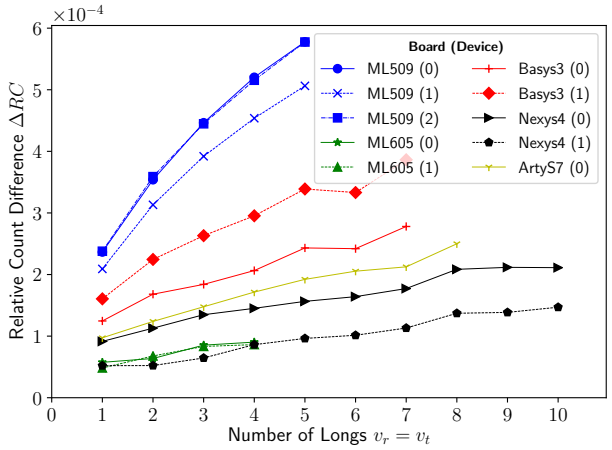


Fig. 10. Relative frequency changes ΔRC for all boards we have tested within different FPGA generations and for different receiver/transmitter overlaps. Results from identical devices are shown in the same color.

The second segment is the region where $v_t \leq v_r$. Here, ΔRC increases linearly with v_t, suggesting that the phenomenon affects the delay of each long wire equally. The final region consists of $v_t > v_r$, where ΔRC remains constant. The reason for this pattern is that there is no additional overlap between the newly added segments of the transmitter and the receiver.

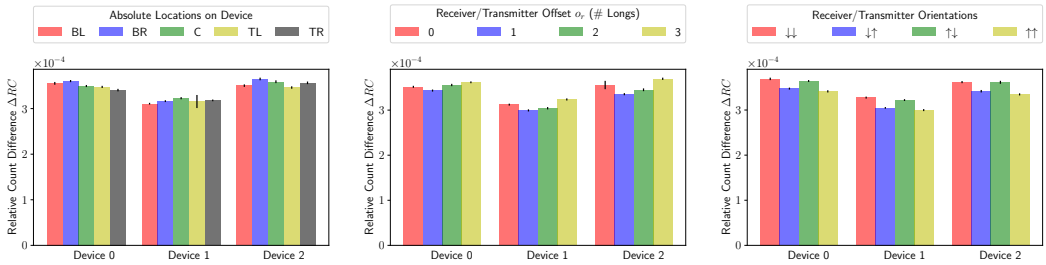
We also identify the effect of a given number of transmitter wires v_t on receivers using a different number of longs v_r. Among receivers with $v_r \geq v_t$, a smaller v_r results in a larger effect. As an example, for $v_t = 3$, the effect for $v_r = 5$ is smaller than it is for $v_r = 3$. This behavior is due to the transmitter affecting only the first v_t out of v_r long wire segments of the ring oscillator. For smaller ring oscillators, these v_t segments represent a larger portion of the number of wires used, and hence of overall delay.

The opposite is true when $v_r \leq v_t$: the larger the RO, the bigger the resulting effect. For instance, for $v_t = 4$, the effect for $v_r = 3$ is larger than the effect for $v_r = 1$. This difference exists because even though the delay of the routing scales linearly, the delay associated with the inverter and buffer LUT stages remains constant. Thus, the routing delay represents a larger fraction of the overall delay (routing delay plus stage delay) for larger ROs. Since this phenomenon only acts on routing delay, larger ROs are affected more than shorter ones.

The trends for all individual boards we have tested with $v_t = v_r$ are shown in Figure 10: all 10 boards we have tested are susceptible to this information leakage on long wires. Figure 10 demonstrates that the leakage is most pronounced in the Virtex 5 family, least in the Virtex 6, and in between for the Series 7 devices. It also shows that there is often variability even within device families and identical boards, highlighting that process variations contribute extensively to the strength of leakage.

8 LOCATION INDEPENDENCE

In order to validate the location independence of the channel, we test three different aspects of the placement of the receiver and the transmitter: the absolute location on the device, the relative offset of the receiver and transmitter, as well as the direction of signal propagation. Figure 11 shows the results for all three experiments on the Virtex 5 devices, with 99% confidence intervals. At a high level, the effect remains approximately constant for each device regardless of the choice of



(a) Absolute Location (b) Relative Offset o_r (c) Direction of Propagation

Fig. 11. Effect of location on the relative frequency of oscillation, with 99% confidence intervals for different placements of the circuit on the device. Absolute location, offset, and signal orientation have little influence on the magnitude of the effect.

parameters. Across devices, the absolute magnitude of the effect varies slightly, but is otherwise almost the same. Any variability across devices is to be expected, since manufacturing variations are known to affect ring oscillator frequencies [11].

Figure 11a shows the results when an identical circuit is placed on different locations of the device: the four corners (bottom/top left/right) and the center. Both transmitter and receiver use 2 longs each, and they are adjacent: when the receiver's location is (x_r, y_r), the transmitter's location is $(x_t, y_t) = (x_r, y_r - 1)$. Within a device, the values are close, and there is no pattern in how the values change between devices. Manufacturing variations within and between devices can thus explain any variability.

The second experiment investigates the effect of the placement of the receiver and the transmitter relative to each other. When the receiver and transmitter have different lengths, it is possible for the two circuits to have the same overlap, but a different starting offset. This relative offset o_r (visually shown in Figure 12) also has minimal effect on the channel. To test this hypothesis, we place a transmitter made up of 5 longs at a fixed location on the device. The receiver, which uses 2 longs, is placed adjacent to the transmitter, but at an offset of o_r full long wires, allowing for four different offset placements. This offset needs to correspond to full long wire lengths due to constraints imposed by the routing architecture of the device. Any other offset would increase the distance d between the transmitter and receiver, which we investigate separately in Section 9. Figure 11b presents the results of this experiment, which show approximately the same consistency both within and between devices as those of the previous experiment.

Note that the relative effect of placing the receiver at various offsets forms a consistent pattern across devices. As an example, the effect for an offset $o_r = 3$ is consistently stronger than it is for $o_r = 1$. This pattern can be explained by the FPGA routing layout: as mentioned in Section 2, the local routing used to get to the various long wire segments is different between each test. Because the local routing resources differ, the ratio between the delay incurred by the long wire segments and the local routing resources changes. As will be discussed in Section 9, while the delay of the long wire segments is affected by the transmitter, the local routing is not.

Using the same setup, and with an offset of $o_r = 2$ full long wires, we change the direction of signal propagation for the transmitter and receiver. In the previous experiments, both signals travelled from the bottom of the device to the top. However, in the Virtex 5 architecture, `VLONG` wires are bi-directional, and can thus propagate signals upwards or downwards. Figure 11c shows the results for the 4 different orientations (receiver and transmitter down, receiver down/transmitter

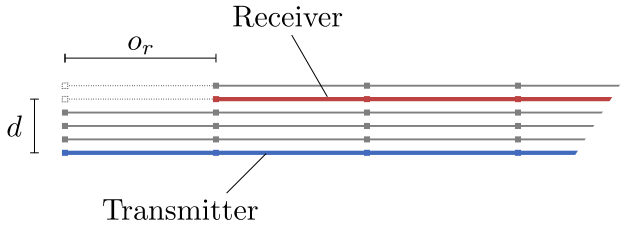


Fig. 12. Relative transmitter and receiver long placement, with respect to distance d and receiver offset o_r.

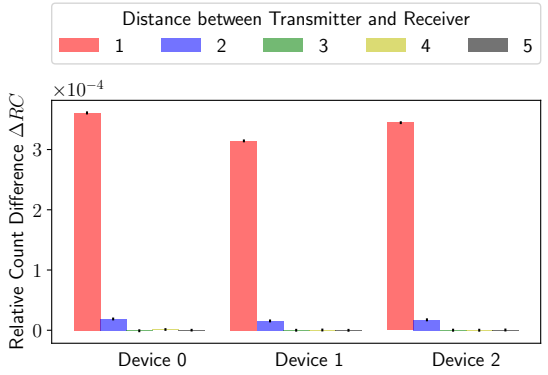


Fig. 13. Effect of the transmitter-receiver distance. Long wires leak information up to two wires away, where distance is defined as in Figure 12.

up, etc.). The relative count difference is approximately the same for all configurations, although as with the previous experiment, we notice a consistent ordering for the four transmission directions across devices. As in the earlier experiment, this pattern can also be explained by the routing layout.

The results of this section illustrate that only long wires need to be manually specified, while registers, LUTs, and local routing can be auto-placed/routed, reducing the complexity of the attack.

9 RESILIENCE TO COUNTERMEASURES

Although we discuss defense mechanisms in more depth in Section 12, in this section we evaluate how close to the transmitter a receiver would have to be in order to decode a message. We do this by varying the distance d (depicted in Figure 12) between the transmitter and the receiver. The results are shown in Figure 13. We see that the phenomenon is still measurable when separating the wires by a distance of $d = 2$, but the effect is 20 times weaker. When the wires are farther apart ($d \geq 3$), there is no correlation between the transmitted and received values, i.e., the data comes from the same distribution according to the Kolmogorov-Smirnov test ($p > 0.75$). In other words, any defensive monitoring must be routed within a distance of two to detect a transmission through the channel, and occupy all 4 wires adjacent to a signal in order to prevent a covert channel from operating successfully, or side-channel leakage from being exploitable.

To test whether an active protection mechanism can disrupt the channel through additional dynamic activity on the device, we measure the strength of the channel in the presence of large, competing circuits which are both in- and out-of-sync with respect to the transmissions. We synthesize 2 large 4096-bit adders, adding different parts of a bitstream produced by a Linear

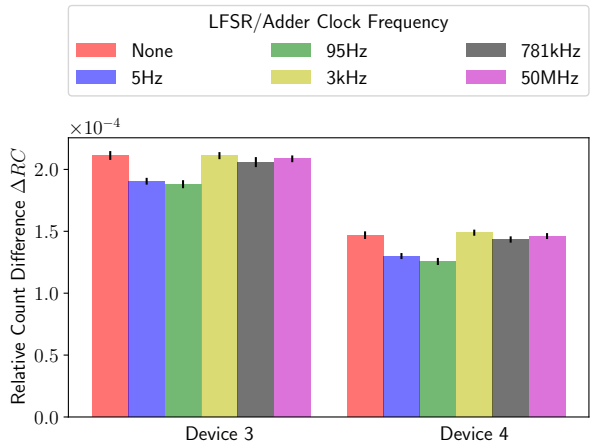


Fig. 14. Effect of activity induced by adders and LFSRs at different clock frequencies. The additional activity has minimal impact on channel quality.

Feedback Shift Register (LFSR). As a result, both the addends and the sums change every time the LFSR produces a new bit. The bits of each sum are then XORed together and drive 2 LEDs for additional current draw. We run the experiment on two Artix 7 Nexys 4 DDR boards, for a transmitter and receiver using 10 longs each.

In order to test transmission and reception under surrounding activity of different switching frequencies, we vary how often the LFSR produces new values by dividing the clock driving it by 2^m, for $m \in \{1, 7, 15, 20, 24\}$, giving us frequencies of 5Hz – 50MHz. The results for the two devices, including the base case of no adders and LFSRs, are summarized in Figure 14, showing that additional activity cannot disrupt the transmissions. However, we note some correlation between the frequency of the activity and the corresponding count difference. The resulting change is not sufficient to hinder transmission, but can be used by the adversary to detect the level of activity on the device, a technique already used by Hardware Trojan detectors [15, 40].

10 SIMULTANEOUS TRANSMISSIONS

In this section we investigate the effect of using multiple transmitters, T_0 and T_1. In our first set of experiments, we return to the Virtex 5 boards, where the transmitters and the receiver use two long wire segments each ($v_t = v_r = 2$). As seen in the timing diagram of Figure 4 (*Transmitter 0* and *Transmitter 1*), the transmitters are driven independently and cycle through all 2-bit combinations over multiple sampling periods. Figure 15 describes the two different transmitter arrangements.

In the first setup, both transmitters are on the same side of the receiver (RTT, where T_0 is at a distance $d = 2$, and T_1 at a distance $d = 1$). Based on the distance discussion of Section 9, we expect only the closest of the two transmitters to have an influence on the ring oscillator frequency. Indeed, Figure 16 shows the ring oscillator counts, which are only affected by the value of the closer transmitting wire, T_1: the data for $T_1 = 0$ and $T_1 = 1$ come from different distributions with $p < 10^{-145}$ according to the Kolmogorov-Smirnov test, while the data is statistically indistinguishable with regards to T_0 ($p > 0.17$). In other words, the effects of the closer transmitting wire overpower the influence of the farther transmitter. This localized eavesdropping capability can be exploited to attack even implementations which balance power usage for security (e.g., dual-rail): an attacker

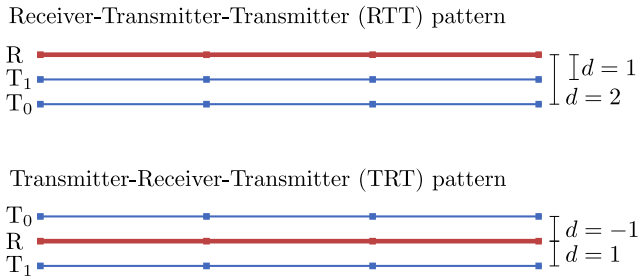
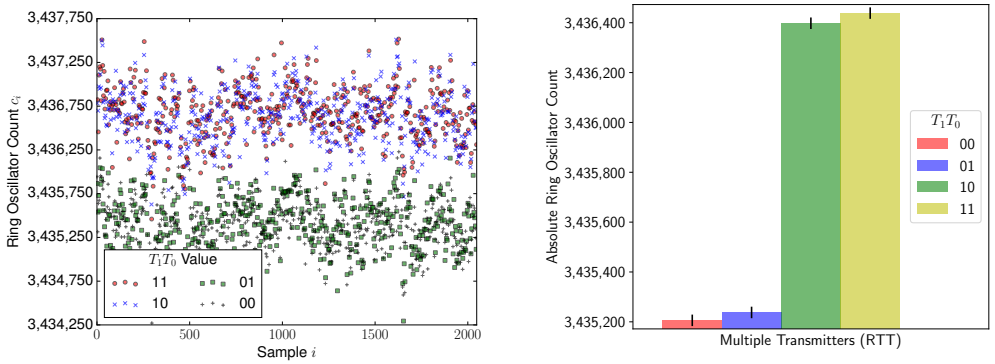


Fig. 15. Relative placement of multiple transmitters and a single receiver.



(a) Raw samples

(b) Average Counts

Fig. 16. Measurements of the ring oscillator when multiple IP cores are transmitting on adjacent wires (receiver-transmitter-transmitter pattern).

can infer information about the state of a balanced circuit without the need for physical access (in contrast to, for instance, Immler et al.'s work [14], which uses external EM probes).

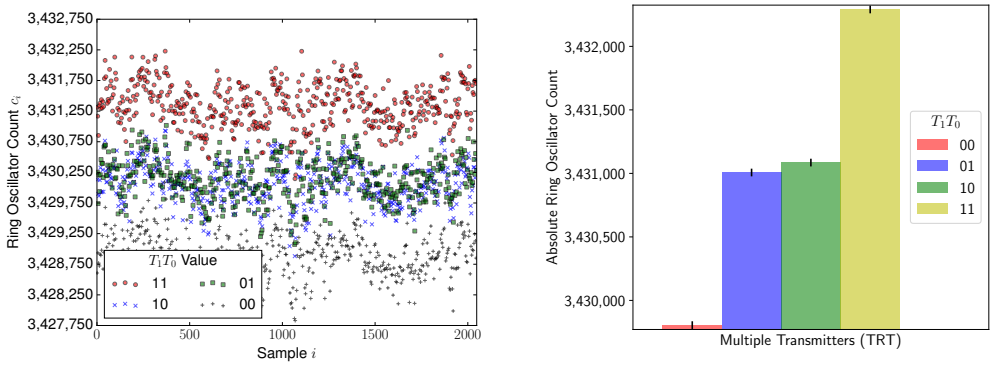
In the second test, the receiver is routed between the two transmitters (TRT, where T_0 is at a distance $d = -1$, and T_1 at $d = 1$), with Figure 17 presenting the RO counts for this experiment. In this setup, T_0 and T_1 have roughly equal influence on the RO frequency. The counts are highest when both transmitters are one, lowest when they are both zero, and in-between otherwise.

In the second set of experiments, we use the Artix 7 boards, and vary the length of the transmitters in the TRT setup. We use the longest receiver possible for both boards, which uses $v_r = 8$ longs for the Basys 3 and $v_r = 10$ for the Nexys 4 DDR. We define the "effective" transmitter length as:

$$v_t^{eff} = b_0 v_t^0 + b_1 v_t^1$$

where transmitter i has length v_t^i and carries value b_i. We expect that for a fixed ring oscillator and a given v_t^{eff}, the ΔRC should remain approximately constant for all possible combinations of $b_i \in \{0, 1\}$ and $0 \leq v_t^i \leq v_t^{eff}$. Indeed, these predictions are validated by our measurements as shown in Figure 18: for a given board, the relative count difference is approximately the same regardless of whether only T_0 (Left), only T_1 (Right), or Both are carrying a logical 1.

Unlike the previous experiments where measurements were conducted using the ChipScope Integrated Logic Analyzer, in this case we control measurements and transfer the data to the PC



(a) Raw samples (b) Average Counts

Fig. 17. Measurements of the ring oscillator when multiple IP cores are transmitting on adjacent wires (transmitter-receiver-transmitter pattern).

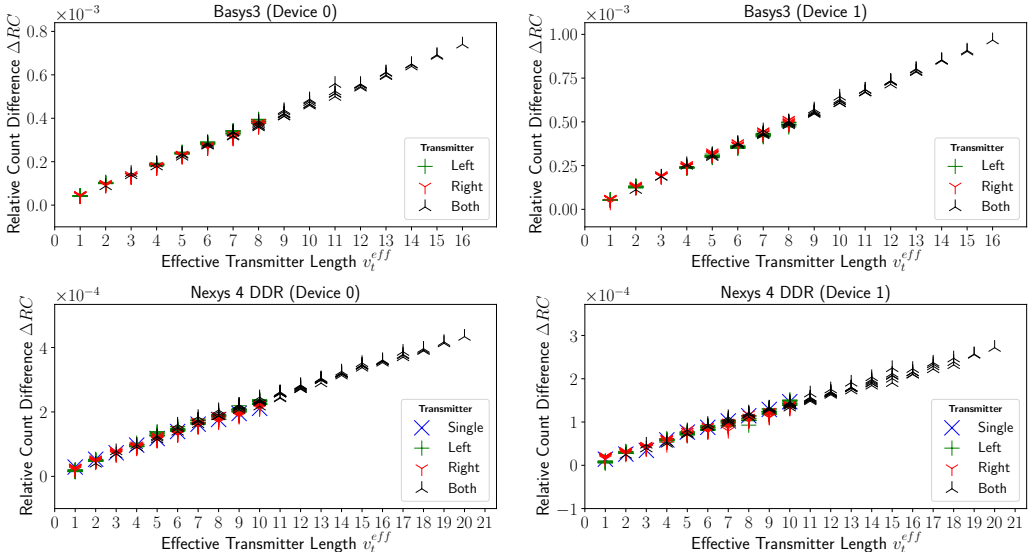


Fig. 18. Relative count difference as a function of different effective transmitter lengths v_t^{eff} for the 4 Artix 7 boards. The strength depends on v_t^{eff} and not individual transmitter lengths.

over the UART. To even further prove that our measurement setup does not have a significant effect on the strength of the phenomenon, we have also plotted the measurements from the experiments of Section 7.2. These measurements had been taken using ChipScope in the single-transmitter (Single) case, and are indistinguishable from our dual-transmitter experiments as expected. When using ChipScope, neither the single-transmitter Basys 3 experiments with $v_r = 8$ nor the dual-transmitter designs for either board could be routed by Vivado, so they are not included for comparison.

The above results suggest that a covert-channel attacker can use dual-transmitters either to increase bandwidth, or to reduce the likelihood of errors in transmissions. In the first case, choosing

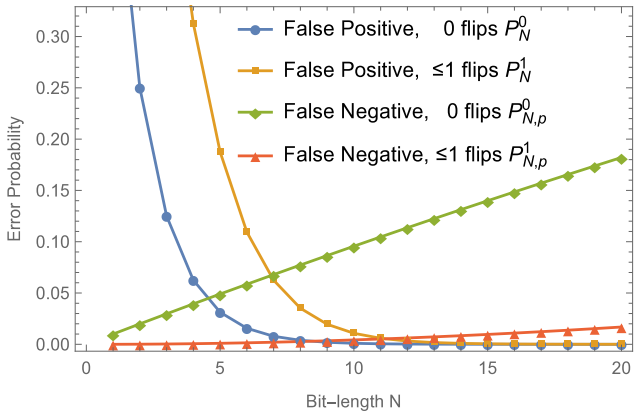


Fig. 19. Probability of error when no transmissions are taking place, and when transmissions are taking place with/without bit-flip error detection for different bit-lengths N and bit-flip probability $p = 0.01$.

transmitters of different lengths allows simultaneous transmission of 2 independent bits, effectively doubling bandwidth, since all four combinations result in different ring oscillator frequencies. In the second case, the transmitters are always in sync, and double the relative effect seen by a fixed ring oscillator. The attacker can thus also use a smaller ring oscillator, which is affected more in relative terms as explained in Section 7.2.

11 EXPLOITING THE LEAKAGE

In this section we discuss exploiting the information leakage from a theoretical perspective. In some cases (such as that of Figure 2), a threshold is sufficient for distinguishing between 0s and 1s, but in other setups (such as that of Figure 5), this separation might not be as clear: the RO frequency may drift due to changes in environmental conditions, such as temperature and voltage variation. We first detail an encoding scheme that enables high-bandwidth covert transmissions (Section 11.1), and then explain how to eavesdrop on dynamic signals through repeated measurements (Section 11.2). We finally perform practical eavesdropping attacks using our theoretical framework in Section 11.3.

11.1 Covert Transmissions

To overcome the hurdle posed by local fluctuations, we propose a Manchester encoding scheme, where 0s are transmitted as the pair $(0, 1)$, and 1s as the pair $(1, 0)$. Since every pair contains each bit once, one can decode the received pair (c_0, c_1) as a 0 if $c_0 < c_1$ and as a 1 otherwise. Using this scheme, transmissions lasting 82 µs using 2 longs as well as transmissions lasting 21 ms using $\frac{1}{3}$ of a long are both recovered with accuracies of 99.0 − 99.9%, without employing any error correction algorithms. Under this encoding scheme, the bandwidth of the channel is $1/(2 \cdot 82 \cdot 10^{-6}) = 6.1$ kbps.

To further distinguish between noise and legitimate transmissions, we can introduce N-bit start-of-frame patterns. Longer N make it harder for "accidental" start-of-frames due to noise when no transmissions are taking place, but they also increase the probability for bit-flip errors during transmissions due to environmental noise and jitter in the ring oscillators. When no transmission is taking place, the probability that $c_0 < c_1$ is $1/2$, i.e., each measurement is equally likely to be interpreted as a 0 or a 1. The probability that noise is interpreted as a start-of-frame when no transmission is taking place is then $P_N^0 = 2^{-N}$. If the probability of a bit-flip error is p (between $0.1 - 1\%$ in our setup), then the probability of an error in the true/intentional start-of-frame pattern is $P_{N,p}^0 = 1 - (1 - p)^N$. These false positive and negative probabilities are shown in Figure 19 for

different N and $p = 0.01$. Even for $N = 5$, the probability of false positives and negatives is 3% and 5% respectively, which is too high for practical purposes.

For this reason, we propose allowing for up to 1 bit error in the pattern. As a result, the number of accepted N-bit start-of-frame patterns is $N + 1$ (the correct one, plus one for every possible position where a bit flip could occur). The false positive rate is therefore $P^1_N = (N + 1)/2^N$, while the false negative rate is $P^1_{N,p} = 1 - (1 - p)^N - Np(1 - p)^{N-1}$, both of which are also shown in Figure 19. Choosing $N = 11$ allows for an almost equal error rate of approximately 0.5%.

It should be noted that when temperature and voltage fluctuations are not random (e.g., if there is a monotonic change in temperature), the probability that $c_0 < c_1$ when no transmission is taking place is not 1/2. In that case, an all 1s (0s) start-of-frame pattern would be triggered easily due to persistently increasing (decreasing) temperatures. An alternating pattern would prevent this from occurring, while a temperature-aware sensor (e.g., a ring oscillator counter normalized for temperature) would allow the covert channel to operate even in these conditions. Different applications could also choose different N based on their bit-flip probability p, and increase the number of allowed bit flips to improve robustness.

11.2 Signal Exfiltration

Signals which are not under the adversary's control may not remain constant throughout the period of measurement. However, as shown in Section 6.2 (Figure 6), the delay of the long wire depends only on the proportion of time for which the nearby wire is carrying a 1, and *not* its switching frequency. This fact reveals the Hamming Weight of the transmission during the measurement period. By repeating measurements with a sliding window, an eavesdropping adversary can fully recover nearby dynamic signals such as cryptographic keys with high probability.

Suppose the adversary wishes to recover an N-bit key K (or, more generally, any internal secret state, such as an AES S-Box input bit), and assume that in one period of measurement, the long wire carries w consecutive bits of the key. We assume that $N = nw$ is an integer multiple of the measuring window w, and refer the reader to Appendix A for details on how to remove this assumption. By making repeated measurements of different but overlapping windows, as shown in Figure 20, the adversary can recover the key with high probability. Specifically, assume the Hamming Weight (measured by the RO count) of the first w key bits K_0 to K_{w-1} (window W_0) is c_0, and that the Hamming Weight of bits K_1 through K_w (window W_1) is c_1. Then, if $c_0 \approx c_1$ (within some device-dependent tolerance), we can conclude that $K_0 = K_w$. If $c_0 > c_1$ then $K_0 = 1$ and $K_w = 0$, while if $c_0 < c_1$ then $K_0 = 0$ and $K_w = 1$. By comparing the next count c_2 to c_1, one can determine the values of K_1 and K_{w+1}, and, more generally, by repeating this process, one can determine the relationship between K_i and K_{i+w}.

Assuming a randomly generated key, the probability that $K_i = K_j$ for $i \neq j$ is 1/2. The probability that all of $S_r = (K_r, K_{w+r}, \ldots, K_{(n-1)w+r})$ are equal is $1/2^{n-1}$, since there are $n - 1$ such pairs. The probability that at least one of the bits in S_r is different than the rest is thus $1 - 1/2^{n-1}$. If at least one is different, we can recover all of these bits. Repeating this argument for all possible remainders $0 \leq r < w$, the probability of recovering the *entire key* is:

$$P = \left(1 - \frac{1}{2^{n-1}}\right)^w \geq 1 - \frac{w}{2^{n-1}} \tag{2}$$

by Bernoulli's inequality. Even if it might appear counter intuitive, the expression shows that longer keys are easier to recover than short keys. A larger window size w relative to the key length makes recovering the key harder as there are fewer measurements over the length of the key. For the same reason, a longer key will increase the recovery probability. This means that asymmetric keys, e.g., those used for signature verification, are relatively easy to recover, as they are typically much

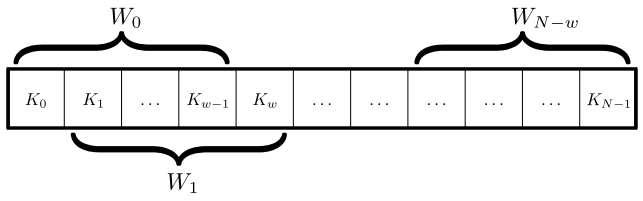


Fig. 20. A sliding window of width w can determine the relationship between key bits K_i and K_{i+w}.

longer than symmetric keys. In the worst case (if the entire key consists of a repetition of its first w bits), our approach reduces the guessing space from 2^N to 2^w possibilities.

In Appendix A we explain how to deal with key lengths which are not multiples of the window size, and how to use multiple window lengths to recover any key pattern with probability 1, or determine that all their bits are equal (i.e., all ones or all zeroes). Moreover, the all 0s and all 1s cases are easy to distinguish as the total RO count for all 1s will be higher than the total count for all 0s.

However, an adversary can reach the same conclusions even with a single window size, provided that the long wire contains deterministic values after the last bit of the key has been transmitted. Essentially, the adversary can consider the w measurements that follow the last key bit as being part of the key itself: $K_N, \ldots, K_{N+w-1}$. There are a few cases to consider after transmitting K_{N-1}:

(1) The long wire contains a fixed 0 or 1 bit, i.e., $K_N = \cdots = K_{N+w-1}$. In this case the adversary can determine K_{N-1} by comparing the count for $K_{N-1} \cdots K_{N+w-2}$ to the count for $K_N \cdots K_{N+w-1}$, and proceed backwards for K_{N-2} and other bits. If at least one of $K_{N-w}, \ldots, K_{N-1}$ is not equal to the fixed value, then the entire key is recovered. Otherwise, the adversary determines that the key consists of all 0s or all 1s.
(2) The HDL code assigns X (don't care) or Z (high-impedance) to the wire. When synthesizing the code, these cases reduce to the case above, something which we have also verified when comparing multiple to single transmitters (Figure 18).
(3) The long wire contains the last key bit K_{N-1}. This case also reduces to (1), but the adversary starts recovery at K_{N-2} instead of K_{N-1}.
(4) The long wire is used to carry other values (such as a key), which would not change on repeated measurements. This effectively increases N, and can be combined with the cases above after the second key has finished transmitting.

The only case where the adversary cannot fully recover the key is when the long wire is updated with a random value that changes at every clock cycle and between measurements. This suggests a possible leakage countermeasure and is discussed in Section 12.3.

11.3 Eavesdropping Attacks

In this section we implement the single-window attack of the previous section on a Basys 3 board. We try to recover an $N = 32$-bit key transferred over long wires, with a window size of $w = 4$, without exploiting any knowledge about the values carried on the long wires after the key has been transmitted. In other words, our attacks represent the worst-case scenario for the attacker. As we show, even this less-sophisticated attacker can recover keys in most setups.

We first "calibrate" our setup by measuring the RO frequency for the all-ones and all-zero keys 1000 times each. This allows us to set a more precise threshold for how to interpret $c_i \approx c_{i+1}$: let the average all-one (all-zero) calibration measurement to be C^1 (C^0), with $\Delta = C^1 - C^0$. We then classify K_i to be the same as K_{i+w} if $|c_i - c_{i+1}| < (1 + \alpha)\Delta$, where α is a relaxation threshold to account for noise (we take $\alpha = 0.5$). Because of the inherent jitter of the ring oscillator, we make

$F = 500$ full passes over the key, collecting $F \cdot (N - w + 1) = 14500$ RO counts, and take the c_i to be the average over the F measurements. In our proof-of-concept, each full pass corresponds to $w = 4$ repetitions of the key (one for each remainder), but an adversary could equally use w parallel counters and require only F passes instead of $F \cdot w$ ones. Similar to the multiple-transmitter experiments of Section 10, we control the setup and transfer measurements to the PC using the UART instead of ChipScope.

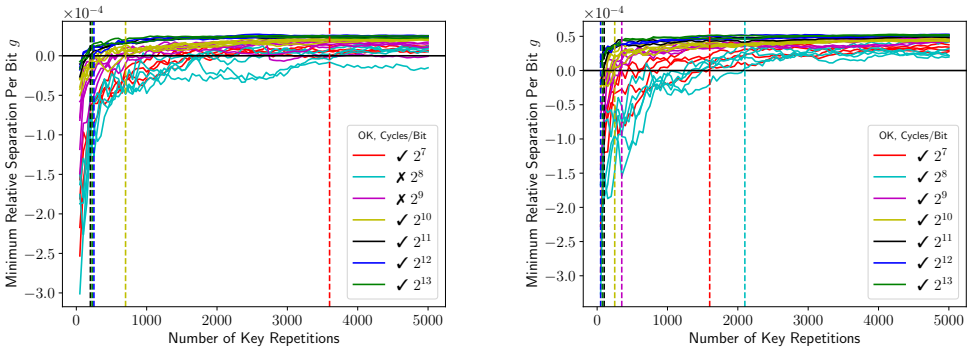
We tested the key recovery of 5 different keys: $\mathbf{K_0} = \texttt{0}$, $\mathbf{K_1} = \texttt{0xffffffff}$, $\mathbf{K_2} = \texttt{0xdeadbeef}$, $\mathbf{K_3} = \texttt{0xd0e1a3f4}$, as well as a randomly chosen key $\mathbf{K_r}$. When the number of long wires used is at least $v \geq 4$, and the number of clock cycles is at least $2^{13} = 82\,\mu s$, our algorithm correctly recovers all 5 different keys. Specifically, $\mathbf{K_0}$ and $\mathbf{K_1}$ are reduced to the $2^w = 16$ possibilities of repeated subkeys explained in the previous section, and $\mathbf{K_2}$ is reduced to the 2 possible values of `X101X110X010X101X011X110X110X111` (the correct value has $X = 1$). $\mathbf{K_3}$ and $\mathbf{K_r}$ are always recovered fully. Reducing the number of longs v or the number of clock ticks reduces accuracy, but still allows partial recovery of keys. For example, with only $2^9 = 512$ clock cycles (5.1 µs), and $v = 2$, our algorithm can correctly recover 30 out of the 32 bits of $\mathbf{K_3}$, and recovers at least 25 bits correctly for all keys when $v \geq 4$.

However, with some additional computation and some more repetitions, we can reduce both the number of longs and the number of clock cycles needed. We tested signals which stay constant for at least 2^7 clock cycles (1.3 µs or 780 kHz), as for faster signals the UART was not fast enough to keep up with the data being produced, and the Basys 3 boards did not have enough memory for larger FIFOs. We found that in 98.4% of measurements with $F = 5,000$ repetitions and $v \geq 1$ longs, the average count for a window is monotonic in its Hamming Weight (HW). In other words, for any two windows of a key with HWs $h_1 > h_2$, the average counts c_1, c_2 for the two windows obey $c_1 > c_2$, making the key recoverable. Put differently, we can recover the key if there are constants $0 = H_0, \ldots, H_w$ such that any window of Hamming Weight h with average RO count c_h satisfies $H_h < c_h < H_{h+1}$. Let K_h denote all subkeys with HW h. Then for two HWs $h_1 > h_2$, we can define the "gap" between them as follows:

$$\text{gap}(h_1, h_2) = \left(\frac{\min\limits_{k \in K_{h_1}} c_k}{\max\limits_{k \in K_{h_2}} c_k} - 1 \right) \cdot \frac{1}{h_1 - h_2}$$

Every window with HW h_1 has a larger average than every window with HW h_2 if and only if $\text{gap}(h_1, h_2) > 0$. Taking the minimum over all HWs, we can interpret $g = \min\limits_{h_1 > h_2} \text{gap}(h_1, h_2)$ as the minimum relative separation per bit if $g > 0$, while $g \leq 0$ suggests that it is impossible to correctly recover all bits of the key. Figure 21 shows g for different keys, measurement periods, and number of measurements collected for $v = 1$ and $v = 2$ longs. For both lengths, 250 measurements are sufficient to recover keys whose bits are kept constant for at least 2^{10} cycles. Although 5,000 measurements are not always enough when using a single long, for the key values we tested, 3,600 measurements can recover keys whose bits change every 2^7 cycles. This number reduces to 2,100 measurements for 2 longs, and is lowered further as the number of longs increases. Our algorithm can thus cope with higher frequencies by either increasing the number of measurements or increasing the overlap between the receiver and the victim wire.

Having calculated these bounds, our algorithm not only reduces a key of size 2^N to 2^w in the worst-case, but it also tells us what the Hamming Weight of the w bits are. Combined with a window of size $w + 1$ or measurements beyond the end of the key, it fully recovers the entire key, even in the all 0s or all 1s case. This sliding window approach allows us to detect much faster signals compared to simple averaging, which would be more susceptible to environmental variations: as



(a) For $v = 1$ longs, some keys cannot be recovered for measurement periods of 2^8 and 2^9 cycles

(b) For $v = 2$ longs, keys are always recoverable

Fig. 21. Number of measurements vs. minimum relative separation for windows of different Hamming Weights. Experiments are repeated for different keys and measurement periods. Vertical lines represent the minimum number of measurements needed to distinguish between different Hamming Weights (i.e., $g > 0$).

noted in Section 7.1, signals which were kept constant for 2^{13} clock cycles resulted in an average absolute count difference of ≈ 4 ticks.

Although clustering measurements in noisy environments is out-of-scope, we also briefly introduce a technique which makes it easier to determine these thresholds without knowing what the key is a priori. To that end, we add a second ring oscillator composed of $v - 1$ long wires, and place it adjacent to the first ring oscillator, but not to the transmitter. Because this secondary RO is only minimally influenced by the transmitted value, we can use it to estimate environmental conditions such as local voltage and temperature. Specifically, during our calibration measurements we make a linear fit for the values of the two ring oscillators during the transmissions of 0s and 1s, and then use this linear relationship to remove the effect of environmental conditions from the measurements which recover the unknown key. With only $F = 500$ measurements and $v = 2$, we were able to perfectly recover 84% of keys tested without any errors, for keys whose bits were kept constant for at least 2^9 clock cycles. Collecting more data points and applying more sophisticated regression techniques would allow recovery of even faster signals, posing a realistic threat even for low-latency applications, such as those found in the finance sector.

12 DISCUSSION

We structure our discussion in three parts: the channel itself (Section 12.1), the cause of the information leakage (Section 12.2), and potential defenses (Section 12.3).

12.1 The Channel

The channel characterized in the previous sections does not require any modifications to the device or special tooling, allowing an adversary to distribute it as IP blocks. The only routing that needs to be specified is the use of the long wires, and the only placement constraint is that the receiver and transmitter longs are adjacent, whether the transmitter is intentional (covert-channel) or unintentional (side-channel). The channel requires very little logic: the entire setup including the signal generation and measurement portions uses just 71 lookup tables (LUTs) and 66 registers, excluding resources to transfer the measurements to a PC for analysis. As an example, our channel

would only use 0.2% of the 33,000 LUTs used in the open-source N200/N210 Universal Software Radio Peripheral (USRP) software-defined radio project [29].

The USRP contains code from Ettus Research, Xilinx, Easics NV, and OpenCores (written by different authors) [30], and illustrates that IP from many organizations can make it into a project. Since third-party code is a necessity, and as modern IP blocks can be quite large, the potential for unintended interaction between different cores increases. An adversary can exploit the routing algorithms, which are forced to route through otherwise monolithic black-box IPs due to resource constraints, enabling his/her blocks to communicate covertly or eavesdrop on nearby signals.

As ring oscillators have legitimate uses, from thermal and device health monitors [6, 42] to Hardware Trojan detectors [15, 40], TRNGs [38], and PUFs [23], the adversary can make dual-use transmitters and receivers. The channel we identify exists whether transmissions are intentional or not, is a threat when an adversary controls one or more IP cores, and can bypass local balancing protection mechanisms as explained in Section 10. Adversarial receivers placed next to third-party logic can therefore exfiltrate secrets carried on long wires. These unintentional long-wire transmissions thus pose new risks for multi-user scenarios, including FPGA/CPU hybrids and cloud infrastructures offering FPGA solutions, which are becoming commonplace.

12.2 Leakage Cause

So far, we have focused on the novelty and applicability of the phenomenon presented, rather than its cause. In Section 6.3, we showed that the phenomenon depends on the use of the long wires, and not the switching activity of circuits, which decreases rather than increases ring oscillator frequency. Gag et al. conducted the only other work which deals with long wires delays, where a RO with a long wire was placed next to other long wires carrying signals which were either equal to the RO signal or opposing it [9]. It was shown that when a nearby long wire has the same value as the RO wire, the frequency of the RO is higher compared to the RO frequency when the nearby long wire has the opposite value (i.e., if the current value on the RO long wire is a 1, the value on the nearby wire is 0 and vice versa). The work by Gag et al. [9] necessitates that the signal of the RO and the nearby wire be in sync, so the wires were directly connected, and static patterns which are independent of the RO signal were not tested. By contrast, in our work, we showed that nearby wires are influenced even when there is no connection between the transmitter and the receiver, and even when the transmitted value remains constant during the measurement period. These two properties can be exploited in constructing a communication channel.

Although Gag et al. [9] broadly categorized their observations as “capacitive crosstalk”, they made no attempt to precisely determine the physical cause behind it. This would indeed be difficult without design information such as physical layout and process-specific parameters. This “lack of electrical detail” on FPGAs is, in fact, well-known and has been identified by multiple authors [1, 2, 8, 28, 34, 39].

As a result, whether the effect we have found exists due to drive-strength issues, electromagnetic emissions, or some other property of FPGAs remains an open question. It is even possible that the wires themselves might not be the cause of the issue, but that the buffers driving them share local connections to the power network. However, without more specialized equipment to x-ray the chips to further narrow down the potential causes, we cannot determine the precise cause, or even whether ASICs would be affected.

Overall, the characterization of the channel is valuable even without access to these details, since we have shown it to always be present and easily measurable on off-the-shelf devices without special modifications. FPGA users cannot alter the electrical behavior of the device, but can only influence how circuits are mapped onto the FPGA. As a result, FPGA circuit designers cannot

change the existence of the channel, and need to be aware of the communication and exfiltration capabilities that this channel introduces.

12.3 Defense Mechanisms & Cloud Deployment

Section 9 showed that one cannot detect transmissions from a distance $d \geq 2$, and that spurious activity (in the form of adders and additional current draw) does not eliminate the transmission channel. Hence, defense mechanisms need to protect a design before it is loaded onto the FPGA. Since long wires are an integral part of the reconfigurable FPGA fabric, detecting the transmitter is not easy: the long can be used as part of the connections within an IP block, carrying sensitive information. Routing algorithms thus need to be modified to account for this information leakage by introducing directives which mark signals (or even entire blocks) as sensitive. The tools then need to add "guard wires", by either leaving the four nearby long wires unoccupied, or by occupying the two adjacent long wires with compiler-generated random signals: occupying and driving all four is unnecessary based on the RTT experiments of Section 10. An example implementation may use the Xilinx Isolation Design Flow which allows physically separating different cores. In fact, this type of physical isolation will prove to be necessary in the data centers of the future, as mere logical isolation and bitstream protection (as suggested by Trimberger and McNeil [37]) are not enough to protect from our attacks. Note that even though this approach will prevent the leakage from occurring, it is particularly taxing for dense designs, and can make placement and routing more time-consuming, or even lead to timing violations. As an example, Vivado could not route our multi-transmitter designs when using ChipScope. Designers using unpatched tools need to be aware of this source of leakage, and must manually look for long wires post-routing, explicitly add guard wires, or, more generally, specify placement and routing constraints for both highly-sensitive signals, and untrusted third-party blocks.

When this is not possible, it becomes necessary to introduce randomness to the sensitive values carried on the long wires. For example, one could make the long wire carry a random bit after every secret bit, and ensure that the last secret bit is also followed with random values that change each time the key is used (Section 11.2). Although these defense mechanisms do not eliminate the leakage (the adversary can still average over large numbers of measurements and get statistical information about the key), they increase the cost of attack and force the attacker to use more sophisticated post-processing techniques.

It should be noted that cloud providers and other FPGA stakeholders may attempt to block the receiver from operating. For example, Amazon EC2 prohibits combinatorial loops such as ring oscillators from their F1 offerings. However, we wish to highlight that the ring oscillator itself is not a fundamental part of the design, and could equally be served by time-to-digital converters (TDCs) or other ways of detecting differences in the delay of the long wires. An alternative RO construction replaces one of the buffer stages with a pass-through latch primitive whose gate and gate enable signals are set to high.[2] We verified on Amazon AWS that this modified RO design bypasses the combinatorial loop protection mechanism, and locally determined that the latch-based RO can still be used to measure long wire leakage. Consequently, simply prohibiting the use of (traditional) ring oscillators is not a sufficient protection mechanism.

Overall, better defense mechanisms for future FPGA generations are needed at the architectural level, and require a deeper understanding of the cause of this phenomenon.

[2]The authors thank Prof. Takeshi Sugawara for this insight.

13 RELATED WORK

Research on side- and covert-channels on FPGAs and other embedded devices has primarily focused on communications between the device and the outside world. Techniques include varying the power consumption of a device and measuring the impulse response [43], changing how much Electromagnetic Interference is emitted by the device [3], or, in the other direction of communication, measuring voltage [31] and temperature changes [35]. These side-channels can be employed in the context of creating Hardware Trojans (HTs) [22], or to watermark circuits [5, 31].

Many of these circuits employ ring oscillators, exploiting their dependence on Process, Voltage, and Temperature (PVT) variations [11]. ROs are primarily used on the receiving end, but they can also be used to transmit information by causing changes in temperature [13]. This technique allows communications between blocks on the same device, under a threat model similar to ours. More recently, Zhao and Suh [41] demonstrated that ring oscillators can be used to recover RSA keys without place-and-route constraints, showing that even physical isolation may not be sufficient to protect against malicious designs in the cloud.

Ring oscillators have also been used in security-sensitive applications, including True Random Number Generators (TRNGs) [38] and Physical Unclonable Functions (PUFs) [23]. Consequently, any mechanism which can be used to manipulate or bias their frequency can also be used to attack these applications. Besides the technique introduced in this paper, prior work has influenced the delays of ROs by altering the power supply [24] and by injecting EM signals [4], resulting in low entropy and cloneability.

As explained in Section 12.2, a switching pattern in sync with the RO's signal increases the RO's oscillation frequency by 1-9% compared to a pattern that opposes it [9]. To achieve this synchronization, however, requires the transmitter to be connected to the output of one of the RO's stages. As a result, as presented, this mechanism cannot be used directly for side-channel communication or to reliably attack the ring oscillator, due to the high accuracy of prediction required for the frequency and phase of the oscillator.

Emphasis has also been placed on using networks of ROs to detect Hardware Trojans on a device [15, 40]. The dynamic power consumed by HTs results in a voltage drop that lowers the RO frequencies compared to those in the Trojan-free "golden" IC, making them detectable. Such prior work depends on the effect switching activity has on the frequency of ROs to detect HTs. As shown in Figure 7, when using short wires, we were able to reproduce the prior effect, where only the number of bit transitions, and not the actual bits themselves, were the cause for the RO frequency drop. However, this no longer holds for long wires: the frequency increases based on the duration for which a 1 is transmitted, irrespective of the dynamic activity. As a result, the channel depends on a fundamentally different phenomenon, which uses the values carried on the wires themselves, and not their transitions.

Prior work [26] has shown that to detect slowly-changing signals, very large circuits (over 14k registers) or long measurement times (2.5h) are needed, in addition to external measurement equipment, and special modifications to the device. By contrast, our work only uses small on-chip circuits, without any special control over voltage or temperature conditions. We can distinguish between the values of signals which remain constant (i.e., have no switching activity) during our period of measurement, which is as low as 82 µs—a measurement period which is also a lower bound for on-chip HT detection using ROs [21].

After we introduced and characterized long wires as a source of exploitable information leakage on Xilinx boards [10], Ramesh et al. [27] repeated some of our experiments on Intel devices, and showed in practice how to recover AES keys that are carried on long wires—an attack which we had initially only covered on a theoretical level. By contrast, this work expanded the characterization of

the leakage by performing measurements on more types of devices (additional Artix 7 boards and a Spartan 7 board), and investigated how simultaneous transmissions can reduce noise and increase the bandwidth of a covert channel. We also introduced a novel way of conducting eavesdropping side-channel attacks which is based on the Hamming Weight of moving windows and also reduces the dependence on voltage and temperature conditions.

14 CONCLUSION

This paper demonstrated the existence of a previously unexplored phenomenon on FPGA devices that causes the delay of long wires to depend on the logical state of nearby long wires, *even when the driven value remains constant*. The effect is small, but surprisingly resilient, and measurable within the device by small circuits even in the presence of environmental noise, and without any modifications to the FPGA. We used this phenomenon to create a communication channel between circuits that are not physically connected, reaching a bandwidth of up to 6 kbps, and an accuracy of 99.9% when using a Manchester encoding scheme in our prototype implementation. The same mechanism can also be used to eavesdrop and recover keys with high probability even when the signals change during the period of measurement. In our proof-of-concept eavesdropping circuit, we recovered signals which are kept constant for as low as 1.3 µs, with an accuracy of more than 98.4%. We showed that the phenomenon is present in four generations of Xilinx FPGAs, and that the channel can be implemented in a variety of arrangements, including different locations, orientations, and with multiple transmitting circuits present. The strength of the phenomenon scales linearly with the number of long wires used, and also dominates a competing effect caused by switching activity. As designs often incorporate circuits from multiple third-parties, this long wire leakage can break separation of privilege between IP cores of different trust levels, or enable communication between distinct cores in multi-user setups. With FPGA and CPU integration becoming more common, and with FPGAs becoming increasingly available on public cloud infrastructures, our work highlights a need to rethink FPGA security in multi-tenant environments.

REFERENCES

[1] Waleed K. Al-Assadi and Sindhu Kakarla. 2008. A BIST Technique for Crosstalk Noise Detection in FPGAs. In *IEEE International Symposium on Defect and Fault Tolerance of VLSI Systems*.

[2] Jason H. Anderson and Farid N. Najm. 2004. Interconnect Capacitance Estimation for FPGAs. In *Asia and South Pacific Design Automation Conference (ASP-DAC)*.

[3] Johannes Bauer, Sebastian Schinzel, Felix Freiling, and Andreas Dewald. 2016. Information Leakage behind the Curtain: Abusing Anti-EMI Features for Covert Communication. In *IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*.

[4] Pierre Bayon, Lilian Bossuet, Alain Aubert, Viktor Fischer, François Poucheret, Bruno Robisson, and Philippe Maurine. 2012. Contactless Electromagnetic Active Attack on Ring Oscillator Based True Random Number Generator. In *International Workshop on Constructive Side-Channel Analysis and Secure Design (COSADE)*.

[5] Georg T. Becker, Markus Kasper, Amir Moradi, and Christof Paar. 2010. Side-Channel Based Watermarks for Integrated Circuits. In *IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*.

[6] Eduardo Boemo and Sergio López-Buedo. 1997. Thermal Monitoring on FPGAs using Ring-Oscillators. In *International Workshop on Field-Programmable Logic and Applications (FPL)*.

[7] Rajat Subhra Chakraborty, Indrasish Saha, Ayan Palchaudhuri, and Gowtham Kumar Naik. 2013. Hardware Trojan Insertion by Direct Modification of FPGA Configuration Bitstream. *IEEE Design & Test (D&T)* 30, 2 (Apr 2013), 45–54.

[8] Thomas De Cnudde, Begül Bilgin, Benedikt Gierlichs, Ventzislav Nikov, Svetla Nikova, and Vincent Rijmen. 2017. Does Coupling Affect the Security of Masked Implementations?. In *International Workshop on Constructive Side-Channel Analysis and Secure Design (COSADE)*.

[9] Martin Gag, Tim Wegner, Ansgar Waschki, and Dirk Timmermann. 2012. Temperature and On-chip Crosstalk Measurement using Ring Oscillators in FPGA. In *IEEE International Symposium on Design and Diagnostics of Electronic Circuits & Systems (DDECS)*.

[10] Ilias Giechaskiel, Kasper B. Rasmussen, and Ken Eguro. 2018. Leaky Wires: Information Leakage and Covert Communication Between FPGA Long Wires. In *ACM Asia Conference on Computer and Communications Security (ASIACCS)*.

[11] Ali Hajimiri, Sotirios Limotyrakis, and Thomas H. Lee. 1999. Jitter and Phase Noise in Ring Oscillators. *IEEE Journal of Solid-State Circuits (JSSC)* 34, 6 (Jun 1999), 790–804.
[12] Ted Huffmire, Brett Brotherton, Timothy Sherwood, Ryan Kastner, Timothy Levin, Thuy D. Nguyen, and Cynthia Irvine. 2008. Managing Security in FPGA-Based Embedded Systems. *IEEE Design & Test of Computers (D&T)* 25, 6 (Nov-Dec 2008), 590–598.
[13] Taras Iakymchuk, Maciej Nikodem, and Krzysztof Kępa. 2011. Temperature-based Covert Channel in FPGA Systems. In *International Workshop on Reconfigurable Communication-Centric Systems-on-Chip (ReCoSoC)*.
[14] Vincent Immler, Robert Specht, and Florian Unterstein. 2017. Your Rails Cannot Hide From Localized EM: How Dual-Rail Logic Fails on FPGAs. In *International Conference on Cryptographic Hardware and Embedded Systems (CHES)*.
[15] Shane Kelly, Xuehui Zhang, Mohammed Tehranipoor, and Andrew Ferraiuolo. 2015. Detecting Hardware Trojans using On-chip Sensors in an ASIC Design. *Journal of Electronic Testing: Theory and Applications (JETTA)* 31, 1 (Feb 2015), 11–26.
[16] Sebastian Korf, Dario Cozzi, Markus Koester, Jens Hagemeyer, Mario Porrmann, Ulrich Rückert, and Marco D. Santambrogio. 2011. Automatic HDL-Based Generation of Homogeneous Hard Macros for FPGAs. In *IEEE International Symposium on Field-Programmable Custom Computing Machines (FCCM)*.
[17] Christian Krieg, Clifford Wolf, and Axel Jantsch. 2016. Malicious LUT: A Stealthy FPGA Trojan Injected and Triggered by the Design Flow. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*.
[18] Christopher Lavin, Brent Nelson, and Brad Hutchings. 2013. Impact of Hard Macro Size on FPGA Clock Rate and Place/Route Time. In *International Conference on Field Programmable Logic and Applications (FPL)*.
[19] Christopher Lavin, Marc Padilla, Subhrashankha Ghosh, Brent Nelson, Brad Hutchings, and Michael Wirthlin. 2010. Using Hard Macros to Reduce FPGA Compilation Time. In *International Conference on Field Programmable Logic and Applications (FPL)*.
[20] Christopher Lavin, Marc Padilla, Jaren Lamprecht, Philip Lundrigan, Brent Nelson, and Brad Hutchings. 2011. HMFlow: Accelerating FPGA Compilation with Hard Macros for Rapid Prototyping. In *IEEE International Symposium on Field-Programmable Custom Computing Machines (FCCM)*.
[21] Maxime Lecomte, Jacques J. A. Fournier, and Philippe Maurine. 2015. Thoroughly Analyzing the Use of Ring Oscillators for On-chip Hardware Trojan Detection. In *International Conference on ReConFigurable Computing and FPGAs (ReConFig)*.
[22] Lang Lin, Markus Kasper, Tim Güneysu, Christof Paar, and Wayne Burleson. 2009. Trojan Side-Channels: Lightweight Hardware Trojans through Side-Channel Engineering. In *International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*.
[23] Abhranil Maiti, Jeff Casarona, Luke McHale, and Patrick Schaumont. 2010. A Large Scale Characterization of RO-PUF. In *IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*.
[24] A. Theodore Markettos and Simon W. Moore. 2009. The Frequency Injection Attack on Ring-Oscillator-Based True Random Number Generators. In *International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*.
[25] Dominik Merli, Frederic Stumpf, and Claudia Eckert. 2010. Improving the Quality of Ring Oscillator PUFs on FPGAs. In *Workshop on Embedded Systems Security (WESS)*.
[26] Amir Moradi. 2014. Side-Channel Leakage through Static Power. In *International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*.
[27] Chethan Ramesh, Shivukumar B. Patil, Siva Nishok Dhanuskodi, George Provelengios, Sébastien Pillement, Daniel Holcomb, and Russell Tessier. 2018. FPGA Side Channel Attacks without Physical Access. In *IEEE International Symposium on Field-Programmable Custom Computing Machines (FCCM)*.
[28] Yajun Ran and Malgorzata Marek-Sadowska. 2003. Crosstalk Noise in FPGAs. In *Design Automation Conference (DAC)*.
[29] Ettus Research. 2016. N200/N210. https://kb.ettus.com/N200/N210. Online; accessed 08 Dec 2018.
[30] Ettus Research. 2018. The USRP Hardware Driver FPGA Repository. https://github.com/EttusResearch/fpga. Online; accessed 08 Dec 2018.
[31] Peter Samarin, Kerstin Lemke-Rust, and Christof Paar. 2016. IP Core Protection using Voltage-Controlled Side-Channel Receivers. In *IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*.
[32] Devu Manikantan Shila, Vivek Venugopalan, and Cameron D. Patterson. 2015. Unraveling the Security Puzzle: A Distributed Framework to Build Trust in FPGAs. In *International Conference on Network and System Security (NSS)*.
[33] Jack R. Smith and Sebastian T. Ventrone. 2011. Multi-processor Chip with Shared FPGA Execution Unit and a Design Structure Thereof. https://www.google.com/patents/US20110307661. US Patent Appl. No. 12/796,990.
[34] Chauchin Su, Yue-Tsang Chen, Mu-Jeng Huang, Gen-Nan Chen, and Chung-Len Lee. 2000. All Digital Built-in Delay and Crosstalk Measurement for On-Chip Buses. In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*.
[35] Ji Sun, Ray Bittner, and Ken Eguro. 2011. FPGA Side-Channel Receivers. In *ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA)*.

[36] Mohammad Tehranipoor and Farinaz Koushanfar. 2010. A Survey of Hardware Trojan Taxonomy and Detection. *IEEE Design & Test of Computers (D&T)* 27, 1 (Jan-Feb 2010), 10–25.
[37] Steve Trimberger and Steve McNeil. 2017. Security of FPGAs in Data Centers. In *IEEE International Verification and Security Workshop (IVSW)*.
[38] Ihor Vasyltsov, Eduard Hambardzumyan, Young-Sik Kim, and Bohdan Karpinskyy. 2008. Fast Digital TRNG Based on Metastable Ring Oscillator. In *International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*.
[39] Steven J. E. Wilton. 2001. A Crosstalk-aware Timing-driven Router for FPGAs. In *ACM/SIGDA International Symposium Field Programmable Gate Arrays (FPGA)*.
[40] Xuehui Zhang and Mohammad Tehranipoor. 2011. RON: An On-Chip Ring Oscillator Network for Hardware Trojan Detection. In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*.
[41] Mark Zhao and G. Edward Suh. 2018. FPGA-Based Remote Power Side-Channel Attacks. In *IEEE Symposium on Security and Privacy (Oakland)*.
[42] Kenneth M. Zick and John P. Hayes. 2012. Low-cost Sensing with Ring Oscillator Arrays for Healthier Reconfigurable Systems. *ACM Transactions on Reconfigurable Technology and Systems (TRETS)* 5, 1 (Mar 2012), 1–26.
[43] Daniel Ziener, Florian Baueregger, and Jürgen Teich. 2010. Using the Power Side Channel of FPGAs for Communication. In *IEEE International Symposium on Field-Programmable Custom Computing Machines (FCCM)*.

A GENERALIZING SIGNAL EXFILTRATION

In this appendix we explain how to remove the assumption that the key size N is a multiple of the window size w, and how to fully recover keys by varying the window size. For $N = nw + m$, with $0 \leq m < w$, the probability that the bits in $S_r = \left(K_r, K_{w+r}, K_{2w+r}, \dots\right)$ are the same is $1/2^n$ for $0 \leq r < m$, since $|S_r| = n + 1$, and $1/2^{n-1}$ for $m \leq r < w$. For $N \geq 2w - 1$, Equation (2) becomes:

$$P = \left(1 - \frac{1}{2^n}\right)^m \left(1 - \frac{1}{2^{n-1}}\right)^{w-m} \tag{3}$$

The lower bound on N is necessary if we wish to recover the first w bits of the key, as we need to have $r + w \leq N$ for each r with $0 \leq r \leq w - 1$ in order to have elements in S_r.

Suppose that the original measurements were not able to recover the bits in S_r because they were all identical. By repeating measurements with a window of size $w + 1$, the algorithm either recovers all bits in the sequence $S'_r = \left(K_r, K_{w+1+r}, K_{2(w+1)+r}, \dots\right)$ or shows that they too are identical.

In the first case, the algorithm recovers K_r, and hence S_r since all its bits are identical. If instead all bits in S'_r are also identical, the entire key consists of a single repeated bit (i.e., all ones or all zeroes). This is because $K_r = K_{w+1+r} = K_{r+1 \pmod w}$, and $K_r = K_{2(w+1)+r} = K_{r+2 \pmod w}$, etc. (one might have to vary r to cover the all the residues mod$w$). The key is thus recovered with probability 1 if there are at least 2 different bits in the key, or it consists of all 0s or all 1s.

A window of size w needs $N - w + 1$ measurements in w runs if using 1 counter (where run r is responsible for $K_{r+w \cdot i}$), or a single run if using w counters. Thus, using both window sizes, and to fully determine all the bits of a key, one needs to take $2N - 2w + 1$ measurements over just $2w + 1$ runs. In other words, the key only needs to be repeated $2w + 1$ times to be fully leaked even when using only one counter.

Received September 2018; revised February 2019; accepted March 2019