Temporal Thermal Covert Channels in Cloud FPGAs

Shanquan Tian and Jakub Szefer
Yale University
{shanquan.tian,jakub.szefer}@yale.edu

ABSTRACT

With increasing interest in Cloud FPGAs, such as Amazon's EC2 F1 instances or Microsoft's Azure with Catapult servers, FPGAs in cloud computing infrastructures can become targets for information leakages via convert channel communication. Cloud FPGAs leverage temporal sharing of the FPGA resources between users. This paper shows that heat generated by one user can be observed by another user who later uses the same FPGA. The covert data transfer can be achieved through simple on-off keying (OOK) and use of multiple FPGA boards in parallel significantly improves data throughput. The new temporal thermal covert channel is demonstrated on Microsoft's Catapult servers with FPGAs running remotely in the Texas Advanced Computing Center (TACC). A number of defenses against the new temporal thermal covert channel are presented at the end of the paper.

CCS CONCEPTS

• **Security and privacy → Side-channel analysis and countermeasures**; *Malicious design modifications*;

KEYWORDS

Covert Channels, Cloud FPGA, Ring Oscillator, FPGA Security

ACM Reference Format:
Shanquan Tian and Jakub Szefer. 2019. Temporal Thermal Covert Channels in Cloud FPGAs. In *The 2019 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA '19), February 24–26, 2019, Seaside, CA, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3289602.3293920

1 INTRODUCTION

FPGAs are increasingly being used in cloud computing infrastructures to allow users to accelerate their computation through use of custom hardware logic. Many cloud providers now support Cloud FPGAs, including Amazon's EC2 F1 instances [6] (for generic use), Microsoft's Azure with Catapult [7] servers (for AI), or Alibaba Cloud's F3 instances [4] (currently "available for testing by invited users"). Cloud FPGAs are also available for remote sharing by academics through deployments such as at the Texas Advanced Computing Center (TACC) [14] (for generic research use).

The business model of Cloud FPGAs focuses on temporal sharing of the hardware between users. When one user is not using the FPGA, it can be assigned to other users. Cloud providers such as Amazon now charge by the minute or even by the second for certain instance types [2], and an FPGA can be almost instantly rededicated to other users once one user is finished using it. In addition to temporal sharing, there is also possibility of spatial sharing. One FPGA can be assigned to multiple users at the same time – this has been explored in academia but so far is not deployed in real Cloud FPGA infrastructures as far as the authors are aware. Interestingly, most researchers have focused on exploring spatial side and covert channels in FPGAs. For example, they have explored cross-talk based channels, e.g., [12]. Meanwhile, channels not requiring spatial proximity or sharing of FPGA have not been widely explored. We present one such new channel, the Temporal Thermal Covert Channel, in this work.

To realize the covert channel, a ring oscillator (RO) heater and RO sensor modules are used. An RO is a temperature-to-frequency transducer suitable for thermal monitoring on FPGAs [3]. With proper calibration, ROs can be used as temperature sensors; comparing oscillation counts of an RO at a fixed location on an FPGA board at different times allows one to observe relative changes in the temperature of the FPGA board over time. Meanwhile, for controlling temperature, a free running RO can be used to generate heat, by constantly toggling transistors and maximizing dynamic power [1]. Use of a sensor diode for measuring temperature is also possible, and potentially more accurate. However, cloud providers can easily disable the access to the diode, or physically remove it to prevent temperature sensing. Use of RO sensors eliminates the need to depend on access to the sensor diode.

In the temporal thermal channel, to transmit information, the sender can either enable RO heater (to generate heat and send 1) or not enable it (to keep FPGA temperature low and send 0). Once the sender vacates the FPGA, a receiver may load their design with RO sensors on the same FPGA to read the current FPGA temperature.

RO based heater and sensor modules were developed for this project and tested both in lab setting and on Texas Advanced Computing Center (TACC) with Microsoft's Catapult servers. With TACC, users can remotely access Altera Stratix V FPGAs which are in the Catapult servers [14], and load their custom logic into the FPGAs. This work shows that without endangering the FPGAs (as first tested on local Stratix V FPGAs) a heater module is able to sufficiently raise the temperature of FPGA that the change can be observed up to 2 minutes later without use of error correcting codes (ECC) for the transmission, and about 3 minutes later when error rates in transmission of the data are less than 30% and can be corrected with use of ECC. This already includes effects of data center cooling which constantly cools the servers and the FPGAs, which cannot be controlled by the attacker as the FPGAs are accessed remotely by the cloud users.

Furthermore, due to abundance of FPGAs in TACC (and likewise in other deployments such as Amazon F1 or Microsoft Azure), the

FPGA '19, February 24–26, 2019, Seaside, CA, USA

ACM ISBN 978-1-4503-6137-8/19/02…$15.00
https://doi.org/10.1145/3289602.3293920

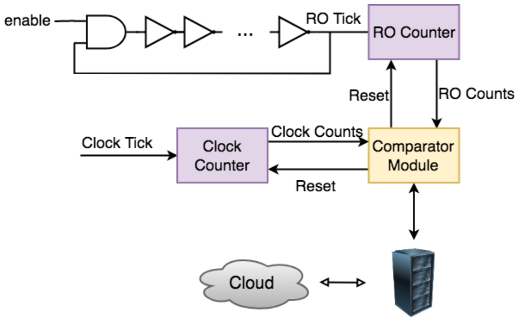

Figure 1: Block diagram of an RO sensor: a free running RO (top-left) drives RO Counter, meanwhile a clock from FPGA's crystal oscillator is used to drive Clock Counter; the outputs from two counters are compared to measure the RO loop's frequency and deduce the temperature; software running on the server attached to the FPGA is used to start and stop the sensor and obtain the RO counts.

heaters and sensors can be run on multiple FPGAs in parallel. The OOK scheme can be used to encode data into heat, and use of multiple FPGAs in parallel linearly increases the bandwidth.

1.1 Contributions to Cloud FPGA Security

This paper makes a number of new contributions:

- Development of the new temporal thermal covert channel based on ring oscillator heaters and sensors.
- Deployment and evaluation of the temporal thermal covert channel on real Cloud FPGA using Catapult servers in TACC.
- Design of defense strategies to mitigate temporal thermal covert channels.

2 BACKGROUND

This section provides background on ring oscillator (RO) circuits on FPGAs. It also gives brief overview of Cloud FPGA deployments that allow for remote access to FPGAs.

2.1 RO Temperature Sensor

A ring oscillator temperature sensor can be built by using an odd number of inverters which are connected in a loop, as shown in Figure 1. The RO sensor further includes an *AND* gate to enable or disable the oscillation of the loop. The sensor works by counting number of oscillations of the loop, compared to some reference counter. The delay through the inverters and wires of the RO depends on the temperature, while the crystal oscillator used for the reference counter is not significantly affected by temperature.

More gates, higher the accuracy, but the lower sensitivity of the sensor to the temperature (T). As T increases, the average inverter gate delay (d) increases, and frequency (f) of the RO oscillations is reduced, the relationship is shown in Equation 1.

$$T \nearrow \Longrightarrow d \nearrow \Longrightarrow f \searrow \tag{1}$$

Specifically, frequency is linearly proportional to temperature. Furthermore, the frequency is inversely proportional to number N of inverter gates in the RO. These relationships are shown in Equation 2 below.

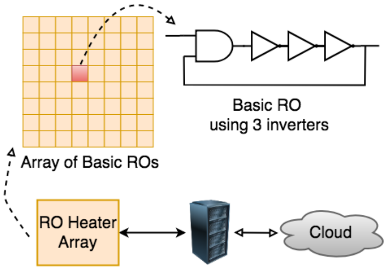

Figure 2: Block diagram of an RO heater: an array of RO loops with 3 inverters each is used; software running on the server attached to the FPGA is used to start and stop the heater.

$$f \propto T \quad and \quad f \propto N^{-1} \tag{2}$$

Based on the design shown in Figure 1, number of RO counts can be used as relative frequency if clock counting time is fixed. Frequency (f) can be replaced by RO counts in Equation 1 and 2.

The RO sensor should be manually placed on the FPGA (via constraints directives in the design tools) so that it is always at same location on the FPGA fabric. Note that the sensor's operation itself will generate heat, thus the sensor should be turned on for as short time as possible so as not to influence the temperature.

The same logical 3-inverter RO placed in different parts of the FPGA, or in same location but on different FPGAs, will give different RO counts for the same temperature. This is due to manufacturing variations as well as different wire lengths that connect different logic blocks on the FPGAs. However, RO placed in the same location on the same FPGA, even after the FPGA is reprogrammed, will have same behavior. And, regardless of the placement, the frequency and temperature keep their relationship, shown in Equations 1 and 2. By reprogramming the FPGA with the same RO in the same location, and measuring the frequency changes, the relative temperature changes can be observed.

2.2 RO Heater

The RO heater is designed as an array of free-running ROs with an odd number of inverters each. An RO heater diagram is shown in Figure 2. Recall from Equation 2 that the frequency (f) is inversely proportional to the size N the RO. Meanwhile, dynamic power (P) is proportional to frequency and number of switching elements. Smaller ROs have higher frequency, but also have smaller number of inverters in the RO. Consequently, an RO with 3 inverters gives the highest power density by allowing most number of ROs to fit in an unit area. The relationship is shown in Equation 3.

$$N \searrow \Longrightarrow \left(\frac{P}{area}\right) \nearrow \tag{3}$$

The RO heater should typically also be manually placed on the FPGA to control the location of the inverters. Especially, the whole RO heater array can be constrained to a pre-specified region of the FPGA to maximize heat density.

2.3 Cloud FPGA Platforms

Increasing number of cloud computing providers are adding FPGAs to their deployments, the so-called Cloud FPGAs. Cloud FPGAs allow users to deploy custom logic into the FPGAs and use them along with other servers and compute resources maintained by the cloud provider. Benefits of Cloud FPGA include on-demand access, and all the tools and licenses are maintained by the cloud provider.

Many existing Cloud FPGA deployments are available today. Amazon's EC2 F1 instances [6] allow access to servers with Xilinx UltraScale+ FPGAs. Microsoft's Azure also includes access to FPGAs and their Catapult servers [7] which Intel (previously Altera) Stratix V FPGAs. Meanwhile, Alibaba Cloud's F3 instances [4] give access to Xilinx VirtexUltraScale+ FPGAs. The Texas Advanced Computing Center (TACC) includes option to access Microsoft's Catapult servers [14] which are the same Catapult servers (and FPGAs) as deployed by Microsoft in their data centers.

The Cloud FPGA deployments typically involve FPGA connected to the server via PCIe bus, and users are able to write software that communicates with the FPGAs via memory mapped registers or through direct memory access (DMA). Users can load their designs on the FPGA, and today each user is dedicated whole FPGA for their use. Spatial sharing does not seem to be deployed in practice, while temporal sharing, explored in this paper, is widely used.

3 DEVELOPING TEMPORAL THERMAL COVERT CHANNEL ON FPGAS

The temporal thermal covert channel on FPGAs is motivated by the fact that Cloud FPGAs are shared in time by different cloud users. While the logical state of the FPGA is erased (or overwritten) when a new bitstream is loaded, the physical state may not be "erased" between when different users use the FPGA. In particular, this work shows that thermal energy of the FPGA takes time to dissipate. In the performed experiments on real Cloud FPGA deployment, the time for the heat to dissipate is on the order of minutes, allowing a user to observe the physical thermal state left over when prior user finished using the FPGA.

3.1 Covert Channel Overview

The timeline of the operation of the new covert channel is shown in Figure 3. It assumes that the sender and receiver share or can access the same set of FPGA boards; and that the sender and receiver are able to load custom logic designs onto the FPGAs. We assume the design will comply with Cloud FPGA provider's design rules and does not violate any Cloud FPGA restrictions[1]. Since multiple FPGAs are used, the sender, Alice, and receiver, Bob, need to pre-agree on which FPGA will be used to send which bit of data.

To transmit information, simple on-off keying (OOK) is used in this project. OOK is a simple form of amplitude modulation, where a logical 1 corresponds to presence of a signal and logical 0 corresponds to absence of a signal. In this work, presence of signal corresponds to high temperature of the FPGA chip, while absence of a signal corresponds to low temperature of FPGA chip. Heating is achieved by use of RO heater. Temperature sensing is achieved by use of RO sensor. Neither operation requires special permissions or hardware, and can be readily executed on today's Cloud FPGA deployments, such as Catapult servers in TACC.

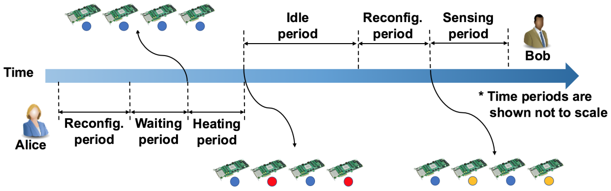

Figure 3: Cloud FPGAs are shared by different users. The timeline shows the six time periods relevant to the Temporal Thermal Covert Channel with 4 FPGAs. Blue dots indicate FPGA is a steady state idle temperature. Red dots indicate the FPGA is at elevated heated temperature. Orange dots indicate the FPGA is still heated above the idle temperature, but heat has dissipated somewhat and it is not as hight as just after heating (red dots). In this example Alice sends 0101 **and Bob is able to receive** 0101**.**

3.2 Threat Model and Assumptions

Our covert channel assumes that the sender, Alice, is in possession of some secret information, such as encryption keys, that she wants to leak to Bob. Further, Alice and Bob can obtain access to the same set of FPGAs in a cloud computing setting and identify each FPGA so they know which FPGA is sending which bit (using unique serial numbers or hardware fingerprinting [13]). The details of how the FPGAs are identified are beyond the scope of this paper. Especially, we assume the *idle period*, discussed shortly, is sufficiently long to identify the FPGAs. Our evaluation shows this is true in TACC.

3.3 Data Transmission with One FPGA

This section first explains how the transmission works with one FPGA, and later section expands on how multiple FPGAs can be used in parallel.

The *reconfiguration period*s (see Figure 3) are needed to account for the time the sender or the receiver take to load their design onto the FPGA. During *reconfiguration period* for Alice she loads the design with the RO heater, while during *reconfiguration period* for Bob he loads the design with the RO sensor.

To begin transmission, the *waiting period* may be needed so the sender, Alice, can ensure that the FPGA is at a steady state temperature corresponding to the FPGA having been unused for a long time. The FPGA is unused at this time to allow to fully cool off, e.g. by loading the RO heater design but not enabling it yet. If Cloud FPGA allows for operations such as explicitly resetting the FPGA, that can be used instead to reset the FPGA and let it stay idle.

Next, the sender enables their actual bitstream (which contains RO heater module), and starts the heating process during the *heating period.* If the sender wants to transmit a logical 1, the FPGA is heated; if the sender wants to transmit a logical 0, the FPGA is not heated and remains at the steady state temperature from the first step. Once the FPGA has been heating for desired amount of time, the sender can log out from the remote machine.

If the required *heating period* is similar to the *waiting period*, i.e. cooling period, the two can actually be overlapped, setting *heating period* to zero. This is because while some FPGAs that are needed

[1] Section 5 discusses defenses cloud providers can deploy, such as special design rules, and how some design checking rules fail to prevent this attack.

to transmit a logical 1 are being heated, the other FPGAs that are needed to transmit a logical 0 are cooling. Our evaluation shows this is actually possible for the TACC scenario.

Following, there is the *idle period* when no user is using the remote machine. The *idle period* assumes the cloud provider enforces that no design is loaded into FPGA between users, e.g., the FPGA is reset after user logs out of the remote machine. During *idle period*, no heating of the FPGA occurs. It is possible that a *zombie period* occurs instead of the *idle period*. Zombie period occurs if the cloud provider allows the FPGA to keep running with existing configuration, even when user logs out. If *zombie period* occurs, then the duration of the *idle period* is effectively zero as the sender's design keeps running and heating the chip up to until the receiver loads their design.

Finally, the receiver loads a new bitstream with the RO sensor during the *sensing period*. After the RO sensor runs and the receiver observes the frequency, the receiver can compute the current temperature of the FPGA chip, and can deduce if a logical 1 (high heat) or logical 0 (low heat) were transmitted by the sender[2].

3.4 Data Transmission with Multiple FPGAs

Using multiple FPGAs in parallel can be used to linearly increase the transmission bandwidth. With Cloud FPGAs, sender can use almost arbitrary number of FPGAs, only limited by their financial cost. Most cloud providers already provide 1 FPGA and 8 FPGA setups by default [6], thus use of multiple of 8 FPGAs seems natural fit for covert transmission.

To transmit the data in parallel, the sender can synchronize the servers with FPGAs, e.g. using network time protocol, and begin transmission at the same time. The sender and receiver need to pre-agree which FPGA will transmit which data bit. Once agreed, the sender starts the transmission by heating selected FPGAs. When *idle period* begins, the receiver needs to locate each FPGA through unique serial numbers, or by use of hardware fingerprinting methods [13]. Once located, the receiver does not have to wait to read all data bits in parallel at same time, but can read each bit individually as soon as each FPGA's temperature is measured with the RO sensor.

3.5 Limits of RO Sensor and Transmission of Multiple Bits per FPGA

Existing work has shown that it is possible to heat up different regions of the FPGA [1]. However, observing the differences in temperature of different parts of the FPGA over time is not trivial. The existing work [1] used high-resolution thermal cameras and observed the FPGA at what is equivalent to 0s of the *idle period*, i.e. at the moment when the FPGA is being heated, and not afterwards.

For this work, we have explored heating up different regions of the FPGA, and indeed, 2 or 4 regions can be possibly distinguished when RO sensor is active at the same time as the heater. However, with non-zero *idle period*, the heat from one region of the FPGA spreads quickly to the whole FPGA and transmission of 2 or more bits per FPGA in the temporal thermal covert channel is not possible in devices we evaluated. As described before, however, multiple bits can trivially be transmitted using multiple FPGAs which are leased from the cloud provider in parallel.

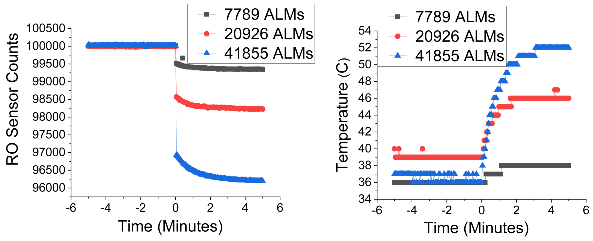

Figure 4: Left figure shows the frequency of the RO sensor as three different sized heaters are turned on at time 0. It can be seen that the heater size determines the maximum temperature reached at different times. Right figure shows the actual temperatures collected with built-in temperature sensor diodes.

4 EVALUATION

The evaluation was performed on Cloud FPGA instances from TACC. The servers in TACC are the same as Microsoft's Catapult servers, and use Altera Stratix V 5SGSMD5H2F35I3L FPGAs. The experiments were performed using heaters of three different sizes, 7789, 20926, and 41855 ALMs[3], corresponding to 4.5%, 12.1% and 24.2% logic utilization out of whole FPGA in the Catapult servers.

4.1 Heating Time Evaluation

Figure 4 (left side) shows the heating time it takes to achieve steady state temperature for the three different heater sizes. The temperature is measured by an RO sensor. Recall, number of RO counts (or RO sensor frequency) decreases as temperature increases, and same RO placed at same location but on different FPGAs will give different RO counts for the same temperature. Thus this graph shows the normalized numbers of RO counts for comparison. Based on the evaluation of the Stratix V FPGAs, for the heaters with 7789, 20926, and 41855 ALMs, it took about 60, 120, and 240 seconds to achieve steady state frequency respectively.

Figure 4 (right side) also shows reference temperature obtained from a built-in temperature sensor which is included in Stratix V. It can be seen the temperature rises much more slowly than the RO frequency. This can be explained by the fact that in Figure 4 (left side) the heater and RO sensor are running at the same time. Therefore, the initial sharp drop of the RO sensor counts is due to FPGA supply voltage change as the large heater array turns on at 0s. The remaining, more gradual frequency reduction after that is due to the thermal processes.

[2] If the receiver has not previously used the RO on this particular FPGA, he will not know what frequency corresponds to the steady state or the heated FPGA. However, he can simply measure the RO frequency, and then wait for the duration equivalent to the sender's *waiting period*, and measure the RO frequency again, knowing the second measurement corresponds to the steady state idle temperature. Comparing the two reveals if the FPGA was in steady state or not during the first measurement. Next time the FPGA is used to transmit a bit of data, the second measurement is no longer needed as the receiver remembers the steady state frequency for the RO on this FPGA.

[3] ALMs (Adaptive Logic Modules) are basic logic cell units in Intel (previously Altra) FPGAs, they are equivalent to Slices in Xilinx FPGAs.

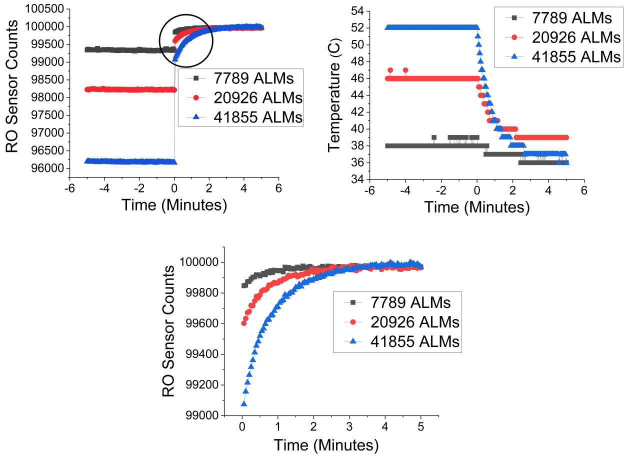

Figure 5: Top-left figure shows the frequency of the RO sensor as three different sized heaters are turned off at time 0. It can be seen that heater size determines the cooling speed and time. Top-right figure shows the actual temperatures collected with built-in temperature sensor diodes. Bottom figure shows the RO sensor frequency after heater is turned off – this is a zoom-in of the top-left figure.

4.2 Cooling Time Evaluation

Figure 5 shows the time it takes for the FPGA to dissipate all the heat generated by the heater. The RO counts (frequency) increase as temperature decreases as the FPGA cools off. Based on the evaluation of the Stratix V FPGAs, for the heaters with 7789, 20926, and 41855 ALMs, it took about 40, 90, and 180 seconds for FPGA to return to idle state temperature. The large initial increase in frequency (Figure 5, top-left, at 0s) is due to FPGA voltage changes as the large heater array turns off. Figure 5 (bottom) shows the zoom-in of the frequency changes after the heater is turned off, showing more gradual increase due to cooling off of the FPGA.

4.3 Bandwidth Evaluation

First, the bandwidth for one FPGA to transmit one bit can be computed using the formula shown in Equation 4.

$$bandwidth = \frac{H(1/2 + e/2) - (1/2) \times H(e)}{((t_r + t_w + t_h) + t_i + (t_r + t_s))} \quad bits/s \qquad (4)$$

where t_w is the waiting time required for sender to wait for the FPGA to cool off, t_h is the heating time, t_i is the idle time between when the sender and receiver are using the FPGAs, t_s is the sensing time, t_r is the reconfiguration time for every new project on FPGA, and e is the error rate. H is binary entropy function where $\mathrm{H}(x) = -x \log_2 x - (1 - x) \log_2(1 - x)$ [5].

Expected bandwidths are computed based on below values. The t_w can be set to 0s as the cooling time and waiting time are about the same (based on data such as from Figures 5 and 4). The t_h can be 240s (based on using heater with 41855 ALMs, from Figure 4). The t_i is variable and depends on the cloud provider setting. The t_r is approximately 4.12 seconds on TACC. The t_s is 30s (time to run a software program on the server and read the RO counts from FPGA on TACC).

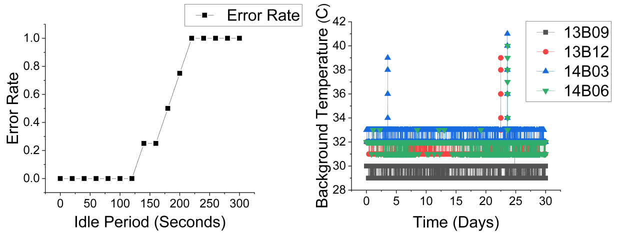

Figure 6: Left figure shows the error rate in the transmission of data using heater of size 41855 ALMs. As the measurement time approaches the cooling period time, the error rate approaches 100%. Right figure shows the ambient TACC temperature as measured using built-in temperature sensor diodes over a period of 30 days, showing steady data center temperature around the FPGAs. 13B09, 13B12, 14B03 and 14B06 are server names.

Bandwidths for using a set of N FPGAs running in parallel to transmit N bits for different idle time periods t_i, are shown in Figure 7, computed using Equation 4 multiplied by N, where $N \in \{1, 8, 16, 32, 64, 128, 256\}$.

4.4 Noise and Error Evaluation

To evaluate the errors in transmission, 8 FPGAs in TACC were used in parallel. Figure 6 (left) shows the data transmission error rate as a function of the length of the idle time between sender and receiver. As expected from the evaluation of the heating and cooling time shown in Figures 4 and 5 data can be successfully transmitted with 100% success rate when the idle time is up to 120s (2 minutes). Many error correction algorithms are able to support error rates up to 30%. If such error correction is deployed, the idle time can be extended to 160s or more.

Noise in the measurements can be influenced by temperature changes in the environment of the FPGA. Figure 6 (right) shows 30-day measurement of the temperature in the TACC FPGAs that were used in this project. The figure shows very steady data center temperature around FPGAs. Thus the noise due to temperature changes in the environment is not factored into this evaluation.

4.5 Total Channel Bandwidth and Cost

Figure 7 shows the possible bandwidth for idle times of 0, 1, 2, 3 and 4 minutes and for 1, 8, 16, 32, 64, 128 and 256 FPGAs used in parallel. The bandwidth is linear in relation to the number of FPGAs used; and the number of FPGAs is only limited by the available FPGAs in cloud provider or the cost that attacker can incur.

To better quantify the cost of leaking a key, an example of leaking 128-bit AES encryption key using 128 FPGAs on Amazon EC2 is shown in Table 1. The cost is surprisingly small and can be used to assign monetary value to the covert channel.

5 DEFENSE STRATEGIES

A number of defense strategies can be deployed by the Cloud FPGA providers to defend against the new covert channel.

Enforce Minimum Idle Period: The best strategy is to enforce a minimum idle period between users, so the FPGA is allowed to cool off to a steady state temperature. Some cloud providers charge by the minute, or even by the second, for use of their resources [2],

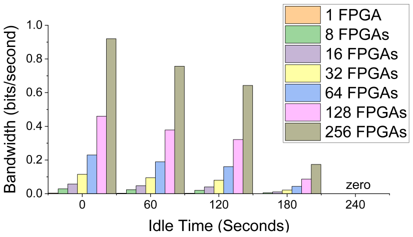

Figure 7: The figure shows the possible bandwidths for different idle times and number of FPGAs used parallel. The bandwidths account for the error rate. Bandwidths are computed using Equation 4 multiplied by the number of FPGAs used in parallel.

Table 1: Estimated cost to leak 128-bit AES key on Amazon EC2. As of September 2018, F1 instances cost $1.65 per hour or $0.0275 per minute. Thus cost for 128-bit key is computed by: 128 instances × time to leak 1 bit per FPGA × $0.0275 per minute. Time is computed using $t_r = 4.12s$ plus $t_w = 0s$ plus $t_h = 240s$ plus t_i (variable and listed in table) plus $t_r = 4.12s$ plus $t_s = 30s$.

AES Key	Idle Period (t_i)	Time to Leak Key	Cost
128 bits	0 s	4.64 min.	16 $
128 bits	30 s	5.13 min.	18 $
128 bits	60 s	5.64 min.	20 $
128 bits	120 s	6.64 min.	23 $

so each idle minute is lost income, but a few minutes of idle period seems reasonable trade-off for eliminating the temporal thermal cover channels. Based on our evaluation, a period of about 10 minutes can fully ensure that FPGA has dissipated all excess heat.

Aggressive Heating or Cooling: A potential alternative is to heat or cool the FPGA to a known temperature before next user is able to use the FPGA. This is a trade-off in shorter idle time, but higher energy cost for the cloud provider to do the heating or cooling. If FPGA is always at a constant known temperature before a user uses it, there is no means for thermal data transfer.

Enact Design Rules to Check Bitstreams: Another defense strategy is to enact design rules that check the FPGA design or bitstream before it is allowed to be loaded onto the FPGA. Already some cloud providers, notably Amazon F1, perform checks on the designs they allow to load. Limitation of the approach is that it is a cat-and-mouse game. For example, the designs could be checked to prevent combinatorial loops from existing in the design, which would prevent RO from being able to be loaded on the FPGA. However, heater or sensor does not have to be based on RO design.

6 RELATED WORK

When two users' designs are loaded onto the same FPGA at the same time, for example, cross-talk between wires on the FPGA chip can be used to steal secrets [12]. Researchers have also explored how partial reconfiguration can be abused when multiple users share the same FPGA to steal another user's IP [9], or reconfigure the FPGA with a malicious bitstream [10].

Non-FPGA research which focuses on temperature includes work on multi-core platforms, where temperature can be used as a covert communication channel [11]. In addition, temperature can also be used to build side channels [8] in smart cards.

To best of our knowledge, this is the first work that explores security issues of temporal sharing of FPGAs and the new temporal covert channel in FPGAs. Furthermore, none of the listed related works have actually evaluated their designs on FPGAs in real cloud computing data centers as we have, rather evaluation is almost always done on local FPGAs "simulating" cloud deployments.

7 CONCLUSION

This paper presented a new temporal thermal covert channel in Cloud FPGAs for covert transmission of sensitive data between users of FPGAs shared in a cloud environment. This work showed cooling of FPGAs in a real cloud deployment such as TACC is slow enough to allow for data transmission through the thermal state of the FPGA chips, and the evaluation showed that even with up to a few minutes of *idle period* it is possible to transmit data. All software and hardware code used in this paper will be made available under open-source license and can be downloaded from http://caslab.csl.yale.edu/code/temporalthermalcc.

ACKNOWLEDGEMENT

The authors would like to thank TACC for the access to their Catapult infrastructure. The authors would also like to thank Dan Holcomb, Rob Moon, Remy Scott, Russell Tessier, and Wenjie Xiong, for their help and/or feedback. This work was made possible by NSF grant 1651945 and FPGA donations from Altera (now Intel).

REFERENCES

[1] Andreas Agne, Hendrik Hangmann, Markus Happe, Marco Platzner, and Christian Plessl. 2014. Seven recipes for setting your FPGA on fire–a cookbook on heat generators. *Microprocessors and Microsystems* 38, 8 (2014), 911–919.

[2] AWS News Blog. 2018. AWS News Blog – Per-Second Billing for EC2 Instances and EBS Volumes. https://aws.amazon.com/blogs/aws/new-per-second-billing-for-ec2-instances-and-ebs-volumes/.

[3] Eduardo Boemo and Sergio López-Buedo. 1997. Thermal monitoring on FPGAs using ring-oscillators. In *International Workshop on Field Programmable Logic and Applications*. Springer, 69–78.

[4] Alibaba Cloud. 2018. Alibaba Cloud – Create an f3 instance. https://www.alibabacloud.com/help/doc-detail/71545.htm.

[5] Thomas M Cover and Joy A Thomas. 2006. Elements of information theory 2nd edition. *Willey-Interscience: NJ* (2006), 187–188.

[6] Amazon F1. 2018. Amazon EC2 F1 Instances. https://aws.amazon.com/ec2/instance-types/f1/.

[7] Microsoft Azure FPGA. 2018. Microsoft Launches FPGA-Powered Machine Learning for Azure Customers. https://www.top500.org/news/microsoft-launches-fpga-powered-machine-learning-for-azure-customers/.

[8] Michael Hutter and Jörn-Marc Schmidt. 2013. The temperature side channel and heating fault attacks. In *International Conference on Smart Card Research and Advanced Applications (2013)*. Springer, 219–235.

[9] Daehee Jang, Hojoon Lee, Minsu Kim, Daehyeok Kim, Daegyeong Kim, and Brent Byunghoon Kang. 2014. Atra: Address translation redirection attack against hardware-based external monitors. In *Proceedings of the Conference on Computer and Communications Security*. ACM, 167–178.

[10] Markus Kucera and Michael Vetter. 2007. FPGA-Rootkits Hiding Malicious Code inside the Hardware. In *Proceedings of the Fifth Intemnational Workshop on Intelligent Solutions in Embedded Systems*. IEEE, 262–272.

[11] Ramya Jayaram Masti, Devendra Rai, Aanjhan Ranganathan, Christian Müller, Lothar Thiele, and Srdjan Capkun. 2015. Thermal Covert Channels on Multi-core Platforms. In *USENIX Security Symposium (2015)*. 865–880.

[12] Chethan Ramesh, Shivukumar B Patil, Siva Nishok Dhanuskodi, George Provelengios, Sebastien Pillement, Daniel Holcomb, and Russell Tessier. 2018. FPGA side channel attacks without physical access. In *International Symposium on Field-Programmable Custom Computing Machines*. 45–52.

[13] G Edward Suh and Srinivas Devadas. 2007. Physical unclonable functions for device authentication and secret key generation. In *Design Automation Conference*. IEEE, 9–14.

[14] TACC. 2018. Catapult - Texas Advanced Computing Center. https://www.tacc.utexas.edu/systems/catapult.