Timing-based side-channel attack and mitigation on PCIe connected distributed embedded systems

Salman Abdul Khaliq
salman.abdul_khaliq@uconn.edu
University of Connecticut

Usman Ali
usman.ali@uconn.edu
University of Connecticut

Omer Khan
omer.khan@uconn.edu
University of Connecticut

***Abstract*—PCIe connected peripheral devices are increasingly deployed in distributed embedded systems. For example, a GPU accelerator connected with a host CPU via PCIe interconnect brings massive performance improvement for artificial intelligence applications. These peripheral devices benefit from shared memory of the host CPU for performance gains, but sharing the host CPU resources brings security challenges. The shared PCIe interconnect hardware of the host CPU can be exploited to create a timing-based information leakage side-channel between multiple connected peripheral devices that are isolated at the software level. This paper proposes an attack setup that consists of GPU and FPGA peripheral devices accessing their data from the host CPU main memory. Both covert-communication and information leakage attacks are demonstrated at a throughput rate of 13.02 *kbps*. A temporal isolation based mitigation scheme is proposed that utilizes time-division multiplexing between the peripheral devices to mitigate the attacks. The paper primarily focuses on demonstrating the security context of the proposed attack and mitigation.**

***Index Terms*—PCIe interconnect, timing side-channel attack, distributed embedded system.**

I. Introduction

Today's exponential increase in computational demand is pushing innovation in distributed embedded systems. Figure 1 shows a distributed embedded system that consists of a host CPU and a set of peripheral devices (i.e., GPU, FPGA, and NIC) attached over PCI-Express (PCI-e) interconnect hardware. The host CPU manages system applications and offloads data for critical computations to the attached peripheral devices for robust execution. For example, an AI accelerator by Google [1] connected over PCI-e interconnect hardware results in 30× performance improvement compared to processing on host CPU alone. The CPU's interconnect hardware allows peripheral devices to directly access memory (i.e., using RDMA or direct cache access using Intel's DDIO technology), resulting in huge performance improvements. However, sharing interconnect hardware infrastructure brings new security challenges.

Peripheral devices share host CPU resources, including caches, main memory, and interconnect hardware (i.e., PCI-e root complex and memory controller). These shared resources are broadly categorized as persistent hardware channels and non-persistent hardware channels. Persistent hardware channels are such channels that store information

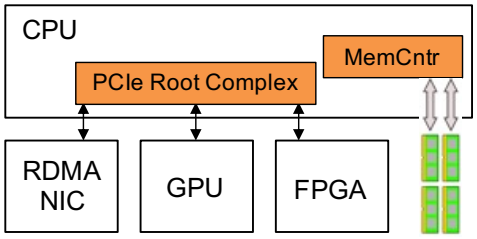

Fig. 1: Distributed embedded system with host CPU and attached peripheral devices.

for a long time or till eviction (i.e., cache and main memories). Contrary, non-persistent hardware channels do not store information for a long time (i.e., execution units and interconnect hardware). This sharing of hardware resources enables a malicious code running on the peripheral device to create performance degradation [2], and timing-based side-channel attacks (SCAs) [3]. Timing-based SCAs are categorized into covert-communication attacks and information leakage attacks. Covert communication attacks enable communication channels between two otherwise unauthorized devices [4]. The information leakage attacks allow a malicious device to extract secrets from another connected peripheral device (i.e., extraction of algorithm secret weights from a machine learning application) [5]. The persistent hardware channels are well studied in secure CPU context [6] [7]. However, non-persistent channels (i.e., interconnect hardware) remains as an unprotected hardware side-channel [8].

Prior work shows that the attackers can target CPU's shared caches [9], and peripheral device shared memory [10] to leak secrets from attached peripheral devices. For example, researchers in [10] are able to demonstrate a covert channel between two clients on NIC devices that exploits the shared memory of a NIC (i.e., a peripheral device) using RDMA technology. Whereas attackers in [9] demonstrate a covert channel attack and key monitoring attack over two attached NIC devices that exploit shared caches of host CPU (i.e., Intel DDIO). A plethora of research on mitigation schemes [6] [7] [11] for persistent channels is pushing adversaries to explore non-persistent hardware channels for timing-based attacks. The memory controllers, a component of interconnect hardware, are exploited in the

context of a single CPU [12], but never explored before in the context of peripheral devices in distributed embedded systems.

This paper focuses on timing-based SCA that targets non-persistent hardware channels, i.e., the shared interconnect hardware in distributed embedded systems. We propose an attack setup that consists of two attached peripheral devices, i.e., a GPU and FPGA connected to a host CPU via PCI-e interconnect hardware. These devices are independent of each other and do not allow sharing of software within each other. The proposed attack exploits the fact that these otherwise virtually isolated peripheral devices share interconnect hardware. This sharing of non-persistent hardware channel create interference in timing variations among these devices. These timing variations are exploited to create timing-based SCAs. To prove the efficacy of attack setup, we demonstrate practical covert-communication and information leakage attacks. Both attacks are demonstrated with a throughput of 13.02 kbps (kilo bits per second).

A practical approach to remove interference to mitigate these attacks is spatial isolation of peripheral devices. For example, only a single peripheral device is allowed to be attached to the host CPU. This approach eliminates SCAs but substantially decreases host CPU capabilities (i.e., only a single peripheral device is allowed). This work proposes a time-division multiplexing (TDM) based mitigation scheme to eliminate timing SCAs. The temporal isolation-based scheme allocates a fixed time slice to each attached peripheral device. This scheme allows multiple devices to connect to the host CPU while eliminating SCAs. The mitigation scheme is implemented on the proposed system setup to demonstrate the successful elimination of the security attacks on shared hardware. This paper leaves the study of performance implications of the mitigation scheme as future work.

II. Related Work

Timing-based SCA is widely used to exploit underlying shared hardware resources to leak sensitive information. Prior work [13] [14] [8] [12] shows that researchers are able to demonstrate covert-channel capabilities and sensitive information leakage with the help of timing-based SCAs. Shared caches [13] [14], TLBs [15], and memory controllers [12] are exploited within context of single CPU in literature. Recently researchers in [10] demonstrated how a malicious peripheral device (i.e., NIC) that shares the SRAM hardware of NIC card is able to leak secrets from co-located peripheral devices with the help of RDMA capability. Similarly, another recent work [9] target shared last-level caches (LLC) of host CPU and leak secrets from a co-located peripheral device with the help of the DDIO technology. For the first time, this paper exploits interconnect hardware, a shared non-persistent hardware side-channel, in the context of peripheral devices in distributed embedded systems.

There is a plethora of research on mitigation of such attacks that exploit shared persistent channel resources (i.e., caches and main memories) in host CPUs. For example, researchers in [11] propose obfuscation-based schemes to eliminate timing differences. Another way to obfuscate is to encrypt the addressing information. Although both approaches introduce high entropy and increase the cost of attack, a motivated attacker can still leak secrets [16]. Contrarily, researchers in [17] [7] [6] propose isolation based schemes. They argue that a practical way to remove the threat of these attacks is to eliminate interference. This paper builds on an isolation-based scheme and proposes time-division multiplexing for interconnected hardware in the host CPU to mitigate timing-based SCAs.

III. PCIe Interconnect Attack & Mitigation

This attack is based on the fact that PCIe interconnect hardware is a limited resource (i.e., the limited bandwidth of 16 GB/s per lane), and sharing of a limited resource among high bandwidth peripheral devices (i.e., an NVIDIA GT 1030 GPU bandwidth demand is 48.06 GB/s) results in resource contention on PCIe interconnect hardware. This contention affects the latency and performance of peripheral devices to perform data-related operations via PCIe. For example, we have empirically observed that a single FPGA device attached to a host CPU using PCIe hardware takes 111 μseconds to read $1KB$ data from the host CPU main memory. However, when two devices (an FPGA and a GPU) are connected to the host CPU, the contention on PCIe hardware causes the FPGA data to read time to increase to 125 μseconds. A timing-based SCA can exploit this timing difference to infer or exfiltrate secret data. Such SCAs are not limited to the FPGA and GPU devices and can be generalized to any PCIe interconnect host and connected peripheral devices.

A. Threat Model

This attack assumes that one or more PCIe connected devices are malicious and are forbidden to communicate with other attached PCIe devices. The malicious PCIe device consists of a counter/timer to measure data movement timings and access to host CPU counters/flags for synchronization purposes.

B. Attack Setup

Figure 2 shows our proposed attack setup, where a GPU (transmitter) and an FPGA (receiver) are connected to a host CPU over PCIe interconnect hardware. FPGA and GPU are allocated spatially isolated memory regions in the host CPU, and both devices are virtually isolated. The GPU is connected to memory module 1, while the FPGA is connected to memory module 4 of the host CPU. With the help of DMA, the host CPU allows peripheral devices to make direct access to the allocated memory regions within the designated memory module. The program on FPGA accesses data from its allocated memory in the host system

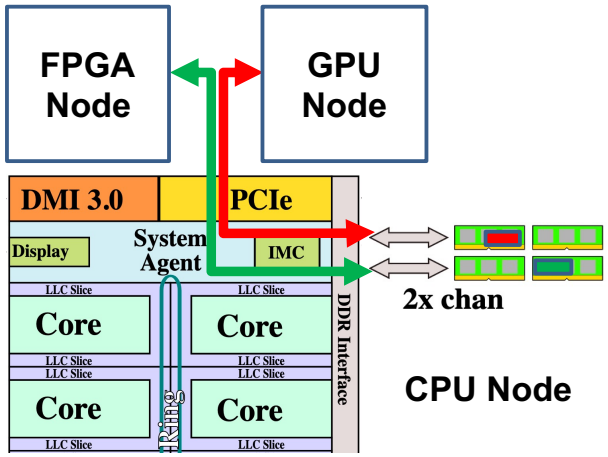

Fig. 2: Logical block diagram of attack setup with a CPU and PCIe connected GPU and FPGA.

and measures the time for each memory read. Similarly, the GPU is programmed to make memory accesses to its allocated memory regions. Although the allocated memory is spatially isolated within the physically isolated memory modules, the host CPU hardware PCIe root complex, System Agent, and Memory Controllers (referred to as PCIe interconnect hardware) are shared. The peripheral devices exploit the shared PCIe interconnect hardware to violate isolation and communicate with each other. For example, the GPU creates contention at PCIe interconnect hardware for t time duration, where FPGA takes a measurement during that t time. The FPGA infers bit 1 if it detects an increase in the latency to read its own data from the memory module. Contrarily, if there is no contention, the FPGA infers a bit 0. A repetition of this procedure is used to transmit a stream of data using this timing-based side-channel.

C. Anatomy of Attack

The proposed attack consists of a host CPU system, a receiver device, and a transmitter device, as shown in Figure 2. The host system consists of processing cores, memory modules, and PCIe interconnect hardware. The transmitter GPU generates *nreplays* number of memory read requests, resulting in contention at PCIe interconnect hardware. It sits idle in case of a no contention scenario. The receiver FPGA takes *nreplays* measurements of data to access its allocated memory from the CPU's main memory. The FPGA infers that either PCIe interconnect hardware is under contention or without contention based on a threshold latency value. If the latency value is below the threshold value, the FPGA infers it as a no contention. Whereas if latency read is larger than the threshold, the FPGA considers PCIe interconnect under contention. This template of contention detection at PCIe interconnect hardware is used to develop practical attacks, i.e., covert communication and information-leakage attacks.

```
// transmitter (i.e., GPU) code

warmup_pcie(bit);
set_sync_bit();

while(sync == 1) {
   if (bit==1){
      // generate contention at host
      for(loop < replay_rate){
         read_data_TLP(@host_address);
      }
   }else if(bit==0){
      //do nothing;
   }
}
```

```
// receiver (i.e., FPGA) code

while(sync == 1) {
   t_start = timer();
      for(loop < replay_rate){
         access_data_TLP (@host_address)
      }
   t_diff = timer() - t_start;

   if (t_diff > threshold){ // infer secret = 1
      secret_bit = 1;
   }else{// infer secret = 0
      secret_bit = 0;
   }
}
```

Fig. 3: The pseudo-code for the PCIe interconnect attack template.

Figure 3 shows a pseudo-code for the attack template. The transmitter code (i.e., GPU) warms up the PCIe interconnect for its intended bit transmission. Once PCIe interconnected is ready for attack, it sets a global synchronization flag[1]. When the synchronization sync flag is set, the transmitter creates contention based on a replay rate or remains idle. On the other side, the receiver (i.e., FPGA) waits for the synchronization sync flag to be set and sets up a timer for timing measurement. It then makes its memory access(es) based on the prescribed replay rate, waits until the data is read, and stops the timer. After taking a measurement, the receiver compares with a predefined threshold and infers contention (secret bit 1) or no contention (secret bit 0).

The threshold is computed offline for a given setup using a statistically significant number of samples with and without contention. The average value of the mean of contention and no-contention distributions is used as a threshold. The PCIe interconnect hardware is shared within multiple devices that induce noise for the attack devices. To overcome this challenge, a replay-based (or repetition) scheme is utilized. The replay is the repetition of transmission and receiving data blocks. For example, a repeated data block three times may result in considerable improvement in correct detection of contention and no contention compared to single data access.

1) Covert-communication attack: Covert-communication attack allows otherwise isolated devices to exfiltrate data from a high-security device to a low-security device. In this attack, the GPU is a malicious transmitter, and the FPGA is a malicious receiver. To transmit a secret bit 1, the GPU makes *nreplays* read accesses to the host memory, which results in contention. Whereas to transmit a secret bit 0, the GPU stays idle and does nothing. The FGPA makes repeated accesses of data blocks from its memory regions in the host CPU at i intervals and measures the time of data block reads. Later based on a threshold, FPGA infers secret bit 1 or secret bit 0. This process is repeated to leaks a stream of secret data.

[1]Another way to achieve synchronization is to use the host CPU cycle counter or independent timers available within peripheral devices, and triggering transmission and receiving at specific times.

2) Information-leakage attack: To demonstrate information leakage attack, this work model a function that process on secret key (RSA square and multiple algorithms is an example of such function) and leaks secret key with the help of timing variations in PCIe interconnect hardware. The GPU is a secure device that processes the function with a secret key. This function consists of function1 and function2. The execution of function1 results in contention at PCIe interconnect hardware, as this function involves data transmission. On the contrary, execution of function2 results in no contention. If the secret key bit value is 0, function2 is executed, whereas if the value of the secret key bit is 1, function1 is executed. Note that the secret being leaked here is the decision variables used to execute function1 only or function2 only. For example, a secret key of 1 will results in a pattern of high contention, whereas if a secret key bit is 0, that results in a pattern of low contention only. This process is repeated to leak the complete secret key.

D. Proposed Mitigation

This paper proposes a time-division multiplexing (TDM) based mitigation scheme for the proposed attacks. The attacks emerge from the latency variations on PCIe interconnect hardware. The demand-based resource allocation of PCIe interconnect hardware results in the contention that enables a malicious device to leak secrets. This demand-based allocation allows a device to occupy all resources, whereas another device is forced to wait for the PCIe interconnect hardware resources. We propose temporal slicing of PCIe interconnect resources, irrespective of demand. For example, if two devices are attached to the host CPU and the allocated time is t μseconds to each device in a repeated pattern. Only a single device is allowed to utilize PCIe interconnect resources during its allocated time slice. This scheme eliminates the timing variations due to the serialization of device activity. Figure 4 shows a timing diagram for the proposed mitigation scheme. For example, in the first slice of t, only GPU can make its requests. The following temporal slice is allocated to the FGPA, where GPU sits idle. This scheme may result in performance degradation but eliminates the attack. This paper shows the security aspect of this mitigation scheme, and the selection of appropriate time for slice and performance impact on real applications is left as future work.

IV. Methodology

This work is implemented on a host CPU Intel Core i7 - 4709 with four 4GB DRAM modules and Ubuntu 20.04 LTS operating system. The NVIDIA GeForce 1030GT with 2GB on-card memory is attached to the onboard PCIe lane. Whereas the Xilinx development board VCU118 with Virtex ®UltraScale+^{TM} FPGA, and 4GB memory module is attached to the host CPU using PCIe hub. The FPGA is programmed with a Xilinx IP *DMA subsystem for PCI express* and accessed using the XDMA driver provided by Xilinx for performance monitoring and debugging. The GPU is programmed with OpenCL library and uses *clEnqueueReadBuffer()* API calls to make read accesses from the host CPU memory.

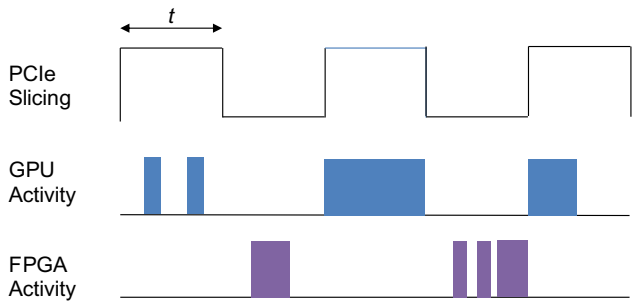

Fig. 4: Timing diagram for TDM based mitigation scheme.

To evaluate the accuracy and reliability of proposed attacks, this work uses True Positive (TP) rate and Discrimination Index (DI) as metrics. The TP rate is the correct identification of the contention, and no contention case at PCIe interconnect hardware. For example, if the transmitter device creates contention, the receiver is able to identify it as a contention case correctly. The TP rate is calculated as:

$$TP = \frac{d_c + d_n}{t_c + t_n} \tag{1}$$

Where d_c is correct detection of contention case, and d_n is the correct detection of no contention case. The t_c is a total transmitted sample of contention, and t_n is a total transmitted sample of no contention.

The DI is a statistical tool that quantifies the difference between two independent distributions. It is calculated based on the statistical mean of each distribution and the variance of each distribution. The DI is calculated as:

$$DI = \frac{\mu_c + \mu_n}{\sqrt{\sigma_c^2 + \sigma_n^2}} \tag{2}$$

Where μ_c is the statistical mean of contention distribution, and μ_n is the statistical mean of no contention distribution. The σ_c^2 and σ_n^2 are the variance of contention and no contention case, respectively.

This work uses a throughput-based metric to evaluate the speed of an attack. The speed is the maximum number of bits transferred over PCIe interconnect SCA in a second using covert-communication or information-leakage attack. The speed metric is represented with bits per second, or *bps*.

V. Evaluation

We have evaluated our attack setup for five configurations of data blocks size in bytes (i.e., n=64, 128, 256, 512, and 1024 bytes) with a replay rate in the range of 1 to 9.

To achieve statistical significance, experiments are repeated for one million samples. Each sample consists of data read of n size for a given replay rate. For example, a data block $n = 64B$ and replay rate of 2, the FPGA makes two 64B data accesses and measures their access time, leading to 2 million memory accesses. Concurrently, the GPU makes the equivalent number of memory accesses to the host CPU memory to create contention.

Figure 5 plots the latency distribution of 1 million samples each for contention and no contention cases with $n = 64B$ and replay rate of 1. The plot shows that the time to read a data block size increases by about 2 μseconds under contention. The distribution is also quantified based on TP rate and DI metrics. Using a threshold value of 15.1974μs, the results show a true positive rate of 92.9% and a 0.4 DI value. Figure 6 plots the distribution for a similar experiment with data black size $n = 1KB$. Using a threshold value of 118μs, the TP rate for this configuration increases to 96.5%, where the DI improves to 1.6.

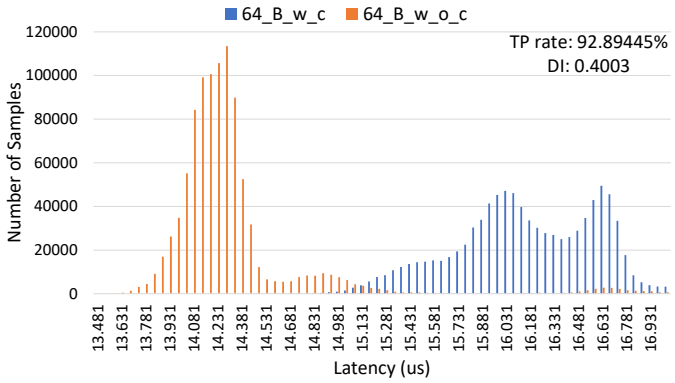

Fig. 5: Latency distribution for 1 million accesses with block size of 64B size, and 1 replay rate.

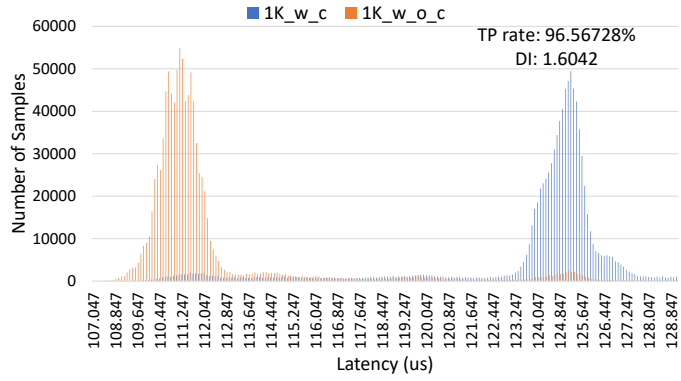

Fig. 6: Latency distribution for 1 million accesses with block size of 1KB size, and 1 replay rate.

An increase in data block size from 64 bytes to 1K bytes (16x increase in size) results in a 3.8% improvement in TP rate and 4× improvement in DI value. Figure 7 plots the TP rate for the five data block size configurations with the replay rate changing from 1 to 9. Although the efficacy of the attack is improved, the increased data size results in a decrease in attack speed. Figure 8 shows a speed plot for the different configurations. The results show that a data block $n = 64B$ with a replay rate of 7 exhibits approximately similar TP rate 96% when compared to $n = 1KB$ with replay rate of 1. But, $n = 64B, replayrate = 7$ configuration shows 2.28× higher speed when compared to the $n = 1KB, replayrate = 1$.

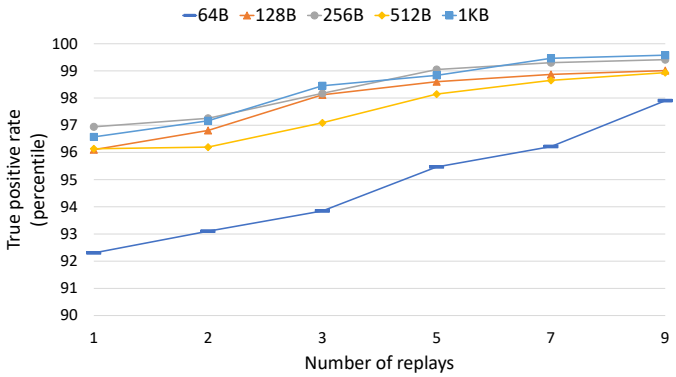

Fig. 7: The TP rate vs replay rate plot with 5 configurations of block size.

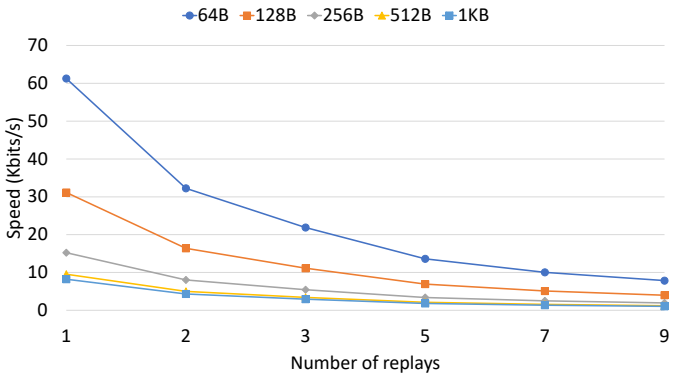

Fig. 8: The speed vs replay rate plot with 5 configurations of block size.

A. *Attacks*

Practical attacks are evaluated with the speed metric. These attacks use a data block size of $n = 64B$, and $replayrate = 5$. A stream of 30 bits is transmitted for a covert-communication attack using the transmitter (i.e., GPU). The receiver makes five accesses of 64B data blocks and measures their commutative time for each bit transmission. Based on the average threshold of 75μs, the receiver infers latency as a bit 0 or a bit 1. This process is repeated multiple times for the stream. Figure 9 represents a time view, where all 30 bits are recovered with a TP rate of ~95% with a speed rate of 13.02 kilo *bps*.

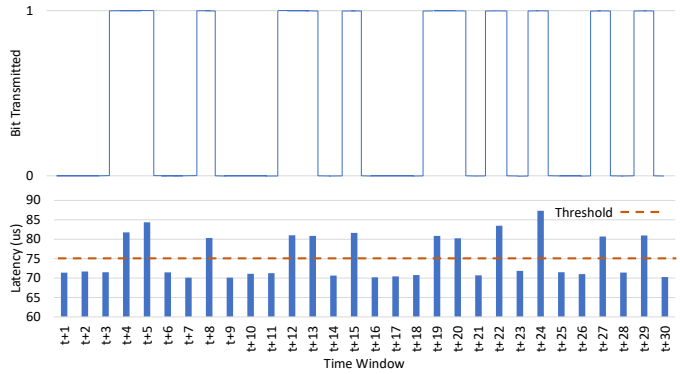

Fig. 9: A time view plot of 30 bits transmission using cover-communication attack.

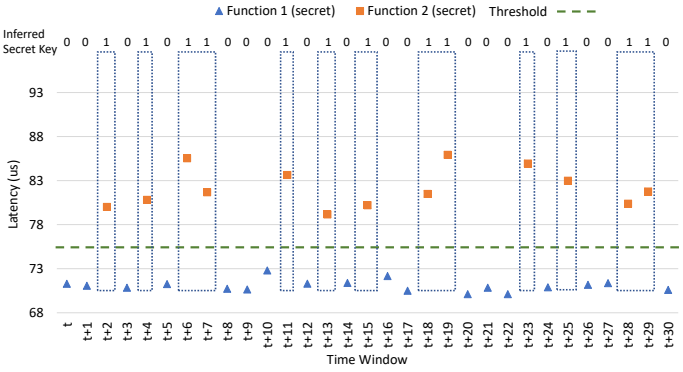

Fig. 10: A time view plot of information leakage attack.

For the information leakage attack, the RSA square and multiple algorithms are modeled. A secret key value is processed using the secure function, where a case of contention is created for secret bit 1 and no contention case for secret bit 0. Figure 10 shows a temporal flow diagram for the attack. The x-axis shows time varying view of function executions, where the y-axis shows latency. The dotted line indicates the latency threshold used for timing variations. The attack recovers the 30 bits with a TP rate of ∼95% using $n = 64B, and replay rate = 5$ with a speed of 13.02 kilo *bps*.

B. Mitigation

The proposed TDM mitigation scheme is evaluated with a t value of 100μs, where each device is temporally assigned 100μs time slots serially to perform operations. Figure 11 shows two latency distributions of one million samples for the contention and no contention scenarios. The distribution shows a negligible latency variation between the two cases. The TP rate metric shows a value of 55.5% and a DI value of 0.01. The TP rate of 55.5% indicates that the receiver fails to correctly identify the transmitted bit as it approaches the statistical probability of 0.5. Whereas the DI value of 0.01 shows a high overlap between the two distributions. These results show that the attack is mitigated.

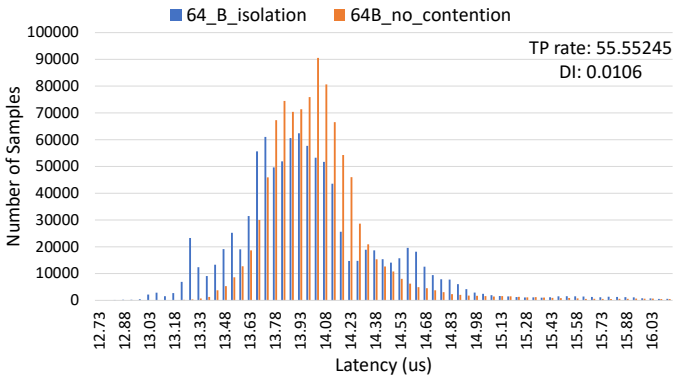

Fig. 11: Timing variations under the proposed temporal isolation scheme.

VI. Conclusion

This paper proposes a timing-based side-channel attack setup that targets the shared PCIe interconnect hardware in distributed embedded systems. The attack setup consists of a host CPU with PCIe attached devices (GPU, FPGA). This paper proved that such a setup could be used to conduct practical covert communication and information leakage attacks. The results show covert communication and information leakage attacks with a speed of 13.02 *kilobps* and ∼95% TP rate. A time-division multiplexing-based scheme is used to mitigate the attacks. The future works aim to evaluate the performance implication of the mitigation scheme for security-critical applications.

VII. Acknowledgments

This research was supported by the National Science Foundation under Grants No. CNS-1929261 and CNS-1916756.

References

[1] T. Norrie, N. Patil, D. H. Yoon, G. Kurian, S. Li, J. Laudon, C. Young, N. P. Jouppi, and D. A. Patterson, "Google's training chips revealed: Tpuv2 and tpuv3.," in *Hot Chips Symposium*, pp. 1–70, 2020.

[2] A. Richter, C. Herber, T. Wild, and A. Herkersdorf, "Denial-of-service attacks on pci passthrough devices: Demonstrating the impact on network-and storage-i/o performance," *Journal of Systems Architecture*, vol. 61, no. 10, pp. 592–599, 2015.

[3] M. Tan, J. Wan, Z. Zhou, and Z. Li, "Invisible probe: Timing attacks with pcie congestion side-channel,"

[4] J. Tian, G. Xiong, Z. Li, and G. Gou, "A survey of key technologies for constructing network covert channel," *Security and Communication Networks*, vol. 2020, 2020.

[5] H. Chabanne, J.-L. Danger, L. Guiga, and U. Kühne, "Side channel attacks for architecture extraction of neural networks," *CAAI Transactions on Intelligence Technology*, vol. 6, no. 1, pp. 3–16, 2021.

[6] H. Omar and O. Khan, "Ironhide: A secure multicore that efficiently mitigates microarchitecture state attacks for interactive applications," in *2020 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pp. 111–122, 2020.

[7] T. Bourgeat, I. Lebedev, A. Wright, S. Zhang, Arvind, and S. Devadas, "Mi6: Secure enclaves in a speculative out-of-order processor," in *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO '52, (New York, NY, USA), p. 42–56, Association for Computing Machinery, 2019.

[8] U. Ali and O. Khan, "ConNOC: A practical timing channel attack on network-on-chip hardware in a multicore processor," in *IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, 2021.

[9] M. Kurth, B. Gras, D. Andriesse, C. Giuffrida, H. Bos, and K. Razavi, "Netcat: Practical cache attacks from the network," in *2020 IEEE Symposium on Security and Privacy (SP)*, pp. 20–38, 2020.

[10] S.-Y. Tsai, M. Payer, and Y. Zhang, "Pythia: Remote oracles for the masses," in *28th USENIX Security Symposium (USENIX Security 19)*, (Santa Clara, CA), pp. 693–710, USENIX Association, Aug. 2019.

[11] M. K. Qureshi, "New attacks and defense for encrypted-address cache," in *2019 ACM/IEEE 46th Annual International Symposium on Computer Architecture (ISCA)*, pp. 360–371, 2019.

[12] Y. Wang, A. Ferraiuolo, and G. E. Suh, "Timing channel protection for a shared memory controller," in *2014 IEEE 20th International Symposium on High Performance Computer Architecture (HPCA)*, pp. 225–236, 2014.

[13] Y. Yarom and K. Falkner, "Flush+reload: A high resolution, low noise, l3 cache side-channel attack," in *23rd USENIX Security Symposium (USENIX Security 14)*, (San Diego, CA), pp. 719–732, USENIX Association, Aug. 2014.

[14] D. Gruss, C. Maurice, K. Wagner, and S. Mangard, "Flush+flush: A fast and stealthy cache attack," in *Proceedings of the 13th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment - Volume 9721*, DIMVA 2016, (Berlin, Heidelberg), p. 279–299, Springer-Verlag, 2016.

[15] B. Gras, K. Razavi, H. Bos, and C. Giuffrida, "Translation leak-aside buffer: Defeating cache side-channel protections with TLB attacks," in *27th USENIX Security Symposium (USENIX Security 18)*, (Baltimore, MD), pp. 955–972, USENIX Association, Aug. 2018.

[16] C. Maurice, M. Weber, M. Schwarz, L. Giner, D. Gruss, C. A. Boano, S. Mangard, and K. Römer, "Hello from the other side: SSH over robust cache covert channels in the cloud," in *24th Annual Network and Distributed System Security Symposium, NDSS 2017, San Diego, California, USA, February 26 - March 1, 2017*, The Internet Society, 2017.

[17] V. Kiriansky, I. Lebedev, S. Amarasinghe, S. Devadas, and J. Emer, "Dawg: A defense against cache timing attacks in speculative execution processors," in *Proceedings of the 51st Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO-51, p. 974–987, IEEE Press, 2018.