ACCESS CONTROL DESIGN AND IMPLEMENTATION FOR DIRECT MEMORY ACCESS ATTACK

by

Marcus Hughes

A thesis submitted to the faculty of
The University of North Carolina at Charlotte
in partial fulfillment of the requirements
for the degree of Master of Science in
Electrical Engineering

Charlotte

2021

Approved by:

____

Dr. Fareena Saqib

____

Dr. Arindam Mukherjee

____

Dr. Kathryn Smith-Weldon

©2021
Marcus Hughes
ALL RIGHTS RESERVED

ABSTRACT

MARCUS HUGHES. Access Control Design and Implementation for Direct Memory Access Attack. (Under the direction of DR. FAREENA SAQIB)

With technology advancements and becoming prevalent in everyday lives, the need for robust and comprehensive security measures is growing. To defend against brute force type of attacks, encryption algorithms have been used to keep user information on hardware safe from outside influences. However, such counter measures are vulnerable to non-invasive side-channel attacks. The direct attacks are used to go after vulnerabilities in algorithm while a side channel attack focuses on analyzing certain physical characteristics of the hardware and using statistical calculations to gain access to private information. In this work, we explore a type of side channel attacks that uses the systems Direct Memory Access (DMA) protocol to compromise the system and later propose a lightweight authentication scheme that identifies the compromised chips and mitigate such attach and makes the system resilient to corrupted hardware without modifications to either the physical hardware or the protocols of the system. The scheme provides protection by generating a unique identifier for trusted hardware and storing the identifiers in a database on the system, allowing only hardware that matches the identifiers to have access to the system and the system's memories.

DEDICATION

I dedicate this paper to my mother Barbara Hughes, my father Marcus Hughes Jr., my sister Jennifer Hughes, and my grandfather Richard Bishop. It was their support that helped me in all things that I do.

ACKNOWLEDGEMENTS

First of all I would like to thank Dr. Fareen Saqib for being my mentor these last couple of years. It was her expertise and guidance that helped me through this process of research.

I would also like to extend my gratitude to Dr. Arindam Mukerjee and Dr. Kathryn Smith-Weldon for being apart of my committee. I greatly appreciate them for giving me the time to review my work and provide feedback.

Finally, I would like to thank the students that I was privileged to work alongside with. Whether they helped me with the research or my studies, I will appreciate them for the rest of my life: Yutian Gui, Chaitanya Mukund Bhure, and Grealdine Shirley Nicholas.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AMBA	Advanced Microcontroller Bus Architecture
BIOS	Basic Input/Output System
CPU	Central Processing Unit
CROI	Correlation-based Region of Interest
DES	Data Encryption Standard
DMA	Direct Memory Access
DROI	Difference-base Region of Interest
FPGA	Field Programmable Gate Array
I/O	Input/Output
ID	Identifier
IOMMU	Input-Output Memory Management Unit
ISA	Industry Standard Architecture
MRd	Memory Read Request
NIC	Network Interface Controller
PCI	Peripheral Component Interconnect
PCIe	Peripheral Component Interconnect Express
PKI	Public Key Infrastructure
PUF	Physical Unclonable Function
ROI	Region of Interest

RPT	Range of Profiling Time
TLP	Transaction Layer Packets
TOFU	A Trust-On-First-Use
TPM	Trusted Platform Module
USB	Universal Serial Bus

CHAPTER 1: INTRODUCTION

One of the important aspects of design is to improve security of the device from invasive and non invasive attacks. In the recent years the cybercrime has increased by almost 600% [1]. There are a wide range of attacks from malware to phishing schemes, as we become more reliant on the internet, the need for security becomes more important. Several attack vectors are possible at lower level of hardware that can bypass higher level of security countermeasures. We are studying hardware security attacks at the architecture level and propose hardware security countermeasures.

Hardware attacks maybe be passive or active attacks to reveal secret information. The passive attacks listen to the data where the device is operating in normal circumstances or the attacker may hijack the operations and introduce the faults to modify the hardware behavior. The attacks may be direct or invasive attacks that tampers a system using techniques such as probing attacks after depackaging the device and reading the signals using probes to readout the memory and registers. The other form of attack is semi-invasive or non-invasive side channel attacks that uses the side channels to break in the system using measurements of how the hardware operate. Protection of one type of attacks does not guarantee security against other types of attacks, for instance, with direct attacks a processor sends the encrypted data to a secure processor and to process the secrets in a tamper-proof, authenticated environment is secure [2] [3]. However, the same method does not work for side channel analysis as it has been shown that using a compromised and older Operating System can still allow an attacker access to the system [4]. One such side channel attack compromises the Direct Memory Access (DMA). DMA is an architectural advancement feature to allow different components of a system to bypass the central processing unit

(CPU) to directly use the system's main memory. Attackers can gain access to the main memory by compromising the DMA and manipulate a systems main memory. There are several schemes to prevent DMA attacks such as key-based authentication and memory virtualization [5] [6].

There are many factors to consider to improve security. The objective of security measures is to have protection against vulnerabilities and attacks from outside forces. The three pillars of cybersecurity are confidentiality, integrity, and availability.

The idea that the user should have the authority to disclose private information to entities of their choosing is the definition of confidentiality. When people think of security, this is probably the most common idea among them, that private data remains private. The theft of private data could cause irreparable harm to users, such as banking information being stolen or the leakage of private industry knowledge. One of the most common methods for attackers to try to break confidentiality is using man in the middle attacks, where they try to intercept private data authorized users are sending between each other.

Integrity refers to keeping the system the way that the user wants it and to have no unauthorized modifications happen without detection. Integrity goes hand in hand with the other pillars of security because if the system has no integrity and can be freely modified by anyone, then confidentiality and availability will be next to impossible to maintain. Some of the more common methods of attacking the integrity include malware, fault injection attacks, and Trojan attacks.

When talking about availability, one might think that it is the same as confidentiality about keeping private information private, but what availability really focuses on is authorized users having access to the system when requested. For instance, if a system has private information that no one, not even the user that put the information in there, can access then it has great confidentiality but if the user cannot access it when they need to, then the system doesn't have availability. Denial of service attacks and ransomware are some of the methods that most attackers use when they want to deny availability. It is the balance of confidentiality, integrity, and availability creates good security for a system.

There are some other security aspects such as an access control policy for a system. An access control policy refers to when a system specifies who can access private resources and when. It is one of the security aspects that enables the confidentiality and protection, as it can keep unauthorized users from private information and modifications as well as making sure that users only have access to the resources they have authorization too.

Authentication is process of granting access to use the host system to a user or component that has been verified as trusted. One of the more widely used forms of authentication is the use of encryption using a special key to identify a trusted user or component that has been assigned to them. A Public Key Infrastructure (PKI) is an example of a key authentication method. Using a private and public key pair, a PKI assigns them to users to identify where the certificate is used to guarantee the public key is assigned to the given user [7][8].

Studies have shown that such security measures provide robust protection, they may also be expensive to incorporate or unfeasible to use in computers that are consumer grade. Even with security measures, some still have vulnerabilities that do not protect direct data transfer [9] [10]. There have also been studies using side channel attacks, specifically power consumption analysis, that show PKI is vulnerable [8].

In this work we propose an access control methodology for DMA compatible PCIe devices using variances in delays as a countermeasure for the DMA attack. The scheme develops a unique identification method that processes measures profiling time of trusted devices and constructs unique identifiers based and store that for secure authentication process. The report is organized as follows: The related work and literature review is discussed in Chapter 2. The attack model and demonstration as well as the proposed scheme are shown in the Chapter 3. Chapter 4 covers the conclusion and Chapter 5 describes future work, that describes possible improvement to explore architectural vulnerabilities and hardware security initiatives.

CHAPTER 2: BACKGROUND RESEARCH

Encryption is widely used in cybersecurity as it ensures the confidentiality and availability of user information by encoding the information so that only the authorized users may know the original message. Encryption is achieved as the information meant to be private, otherwise known as plaintext, is processed through various algorithms and becomes an unreadable ciphertext. To understand what the original plaintext was from the ciphertext, an authorize user has a key to decrypt the text and read the original information. An encryption is robust if there is no way to decrypt the information without the authorized key.

Direct attacks are commonly used against encryption, the attacker using various values against the target to receive responses from the encryption with the goal to calculate the algorithms used in the encryption process and determining the private information. The industry standard for encryption is called Data Encryption Standard (DES) and was even used by the government for their encryption of data. As technology and security measures advanced, it also allowed for more powerful CPUs to be able to analyze a DES encrypted ciphertext and extract the key in less than a day [11].

2.1 Direct Memory Access (DMA)

The traditional way for transferring data to and from a host machine's main memory to an external peripheral component is via the CPU, usually needing several cycles to do so. This is because the CPU must read every block of data that passes through it, causing a significant number of resources to be used and impacting the performance of the system. A way developed to counteract this resource heavy process is by implementation of DMA. DMA allows the peripheral device's read and write requests to sidestep the CPU and independently read and write to the main memory. Without the need of the CPU, the performance of the system increases overall to allow the CPU to be used for other tasks.

Several different types of bus architectures that can support DMA such as Advanced Microcontroller Bus Architecture (AMBA), Industry Standard Architecture (ISA), and Peripheral Component Interconnect (PCI). To perform the process of DMA and run efficiently, a system needs to have a DMA controller. The basic function of the DMA controller is to transfer data to different parts of the system to one another, taking the place of the CPU in data transmission, doing this by using different buses and channels. When the DMA controller initializes, the memory controller initiates the memory read or write cycles, depending on what is needed at the time, as well as provide memory addresses for data transfer. When the process is complete an interrupt is sent to the CPU to stop data transmission.

2.1.1 DMA Controller

In the host machine, a control unit on the I/O interface contains the DMA controller. The controller receives a signal for a DMA request and then controller the bus and the signal lines so that the component that sent the signal in the first place can read or write the data to the main memory [12]. One example of a control unit is the Intel 8237 which is specifically a DMA controller with four-channels, meaning that it can store the DMA information in it at once [13]. The data that comprises the DMA information is all about the transfer meaning the memory address that is related to it, whether it is reading or writing to that memory, how large the data transfer is, as well as the status of any ongoing transfers through the controller.

2.1.2 DMA Memory Access Protocol

DMA helps improve the efficiency of Peripheral Component Interconnect (PCI) devices and Peripheral Component Interconnect Express (PCIe) devices. When one of these devices are connect and powered by the host machine, they usually remain inactive until they need to respond to configuration transactions that usually deal with reading the device's identifiers, such as the vendor ID and the device ID. When a host machine with PCI/PCIe devices starts its booting process, the devices are offered access by the BIOS/kernel and then are allocated both address space and I/O regions. Through this process, when a device driver goes to access the PCI/PCIe device, the previously allocated addresses and I/O regions for a device are already mapped to the address space of the CPU [14].

There are 2 different ways that DMA transfers data from the main memory to the peripheral device, one way is triggered by the process call and then other is triggered by data acquisition devices. Figure 2.1 demonstrates the different processes associated with the different ways of triggering DMA. Generating and allocating a DMA buffer can also be completed two ways, either through coherent DMA mapping using *dma_alloc_coherent* and *dma_free_coherent* or through streaming mapping with use of *dma_map_single* and *dma_unmap_single* [12].

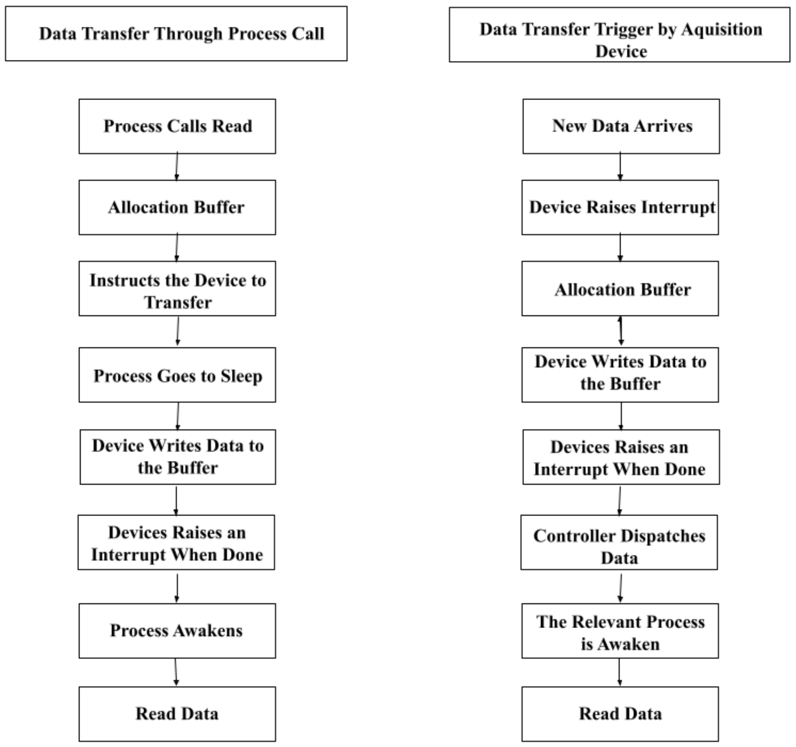

Data Transfer Through Process Call
Process Calls Read
Allocation Buffer
Instructs the Device to Transfer
Process Goes to Sleep
Device Writes Data to the Buffer
Devices Raises an Interrupt When Done
Process Awakens
Read Data
Data Transfer Trigger by Aquisition Device
New Data Arrives
Device Raises Interrupt
Allocation Buffer
Device Writes Data to the Buffer
Devices Raises an Interrupt When Done
Controller Dispatches Data
The Relevant Process is Awaken
Read Data

Figure 2.1: The two processes of data transfer used by DMA.

2.1.3 PCI/PCIe Devices

PCI and PCIe are both buses that connect an external peripheral device to the host machince, the difference between them is that PCI is a parallel connection while PCIe is a serial connection, making it faster than PCI [15]. The way that PCI and PCIe devices are configured gives them several different identifiers. Vendor ID and device ID are used to identify the manufacturer and the device itself. Another set of identifiers, Subsystem Vendor ID and Subsystem Device ID are used by the system to tell similar devices apart. Figure 2.2 demonstrates the typical configuration space of a PCI/PCIe device. The struct pci_driver is used to structure a PCI driver, which is made of several variable and function callbacks that let the PCI core to understand the PCI driver [12].

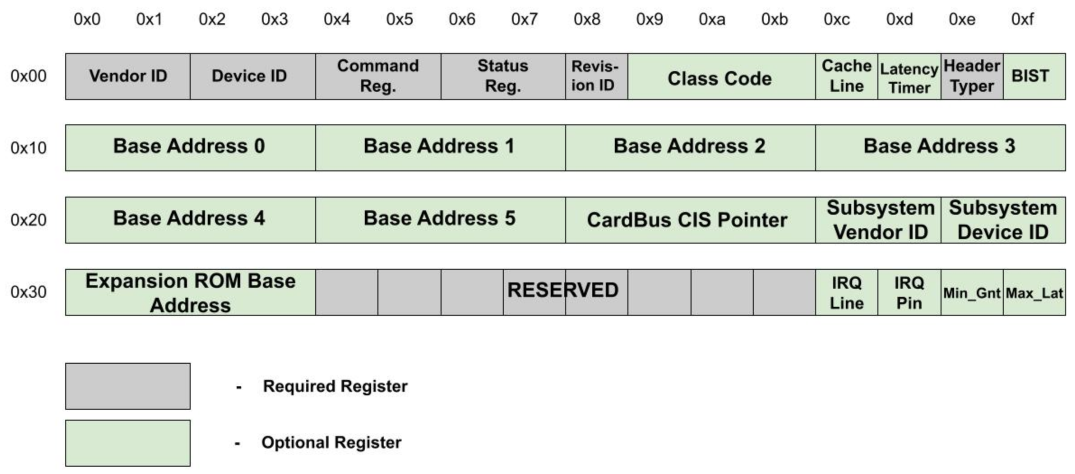


Figure 2.2: Memory configuration of a PCI/PCIe device.

Another aspect to take into account is the latency of the PCI/PCIe bridge itself. When PCIe was first introduced, studies showed PCIe had a negative impact on performance [16]. As the state of art evolved and the landscape of computer architecture changed, PCIe has become one of the more predominant architectures for I/O connectivity. This is in part that when transitioning to a new generation, PCIe has been about to double its bandwidth, just recently the fifth iteration of PCIe was able to achieve 32.0 GT/s while the sixth iteration is able to achieve GT/s [17]. Though PCIe has adapted and has become more efficient, it is still considered to be a bottle neck for throughput in some applications. When working with a NIC, studies have show that PCIe is responsible for 77% of latency for packets measuring 1500 B and has shown that the percentage increases as the packet size decreases, up to 90% of the latency for small packets is due to PCIe [18].

2.1.4 DMA Attacks

While DMA transfer does improve efficiency with the CPU needing to perform less cycles, it does present new avenues of vulnerability, allowing access directly to the main memory of the system. With vulnerable hardware, attackers can gain access to a victim machine's memory and be able to read and write to it easily. In one such case with a Firewire vulnerability, it was shown that an attacker could take control of the DMA controller and gain memory access [19].

With the use of a program called PCILeech an attacker can use a PCIe device to connect to a victim machine and can read and write the memory without for hardware drivers and can run on a variety of hardware and OS [20]. With PCILeech, attackers can bypass password requirements, load unauthorized drivers, have the victim machine execute code, and can create system shells in the victim machine [21]. Using the Montgomery's ladder technique can cause privacy leakage from the main, obtaining secret bits from a 128-byte message with a 64-byte secret exponent [22].

The Intel Advance Threat Research team has also shown that, by modifying a WiGig dock, they could compromise a laptop by connecting to the dock and perform a DMA attack wirelessly and dump the victim machine's memory remotely [23] [24]. There has also been evidence to shoe that the Thunderbolt protocol with access-control enabled is vulnerable to DMA attacks, an attacker being able to use identity clone and spoofing to gain access to the main memory through the Thunderbolt port [9].

2.2 Preventative Measures

One way to mitigate DMA attacks has been developed by Microsoft called BitLocker, introduced in Windows Vista. BitLocker is a pre-boot authentication process, performing a full-disk encryption and storing the keys that are generated on to a Trusted Platform Module (TPM) [5]. During pre-boot, the user provides a unique identifier, either a specific pin or even a USB startup key, that tells BitLocker to access the encryption key and store it to memory. While BitLocker does add more security, it only provides encryption for hard drives and removable data drives, along with the encryption and decryption processes causing high time overhead. The system can also still be compromised through a DMA attack when it finishes the booting process.

In Windows 10 version 1709, Microsoft created new features to mitigate BitLocker's flaws by blocking hot-pluggable PCI/PCIe ports from the main machine until a trusted, registered user signed into the system [5]. This method helps with vulnerabilities after booting as well as helping to secure the keys generated by BitLocker from compromised hardware. This approach does present an issue in that a user will not be able to use PCI/PCIe ports if the system remains locked.

One way to help mitigate the effects of a DMA attack is to make sure that DMA access is limited to only certain parts of the memory. Intel and AMD both created their own versions of this method, Intel creating VT-d while AMD designed AMD-Vi, though both can be referred more generally as Input-Output Memory Management Unit (IOMMU) [6] [25]. For a IOMMU to restrict DMA to only certain parts of the memory, it forms a connection from the DMA bus to the main memory, converting the IO virtual addresses to physical addresses, ensuring that each DMA access are not allowed to the whole memory. IOMMU does have a harder time protecting systems with smaller memory sizes, as there is only so much space to allocate to DMA accesses. It has also been shown that with the IOMMU enabled, the throughput of the DMA process drops significantly, seeing a drop of 70% for 64B DMA and a 30% drop from 256 DMA [18]

A Trust-On-First-Use (TOFU) as well as encryption-base authentication for access control can help mitigate the effects of DMA attacks [26]. In encryption-based authentication the host machine and each device used in it can have a key-based certification, however at any time all components must be able to be authenticated which could make such a scheme costly. When using a TOFU based scheme the system records the identifiers for a trusted device when it is first connected to the host machine. The identifiers for the trusted devices are stored onto a database on the host machine, any device not recognized in the database are blocked from accessing the system. There is one flaw with this process as PCIe devices do not have unique identifiers generated for them, their Vendor ID, device ID, and the slots ID being the only thing recorded into the database. This means that if an attacker were to replace a device with a compromise one of the same model and manufacture into the same slot, the trusted database would not see it as a new device being connected to the system.

CHAPTER 3: EXPERIMENT

3.1 Attack Model

As mentioned in Chapter 2, PCIe is a high-speed serial bus for connecting peripheral devices to the main system to transfer data. What makes a DMA attack possible is through a flaw of the uppermost layer of PCIe when communicating with the host. The device and host exchange Transaction Layer Packets (TLP) through this layer, however the layer itself does not provide any security for data privacy. To demonstrate threats to security like PCILeech, an attack model was created referencing the work in [20][21].

The basic flow of the attack is that a compromised PCIe device is installed to a victim machine and can be connected to the attacker machine using either USB or Ethernet. When the attacker wants to have access to the main memory, the attacker machine sends a command to the compromised device to send a TLP for a Memory Read Request (MRd). When the MRd is read by the victim machine, the victim machine sends back data to the compromised device in another TLP. When the compromised device has the data from the memory, the attacker machine can dump the data from the compromised device received from the memory can even search for specific memory addresses in the data.

In order to perform and demonstrate a DMA attack, the components were needed: a victim machine, an attacker machine, and a compromised PCIe device. The PCIe compatible device used in this experiment was a NeTV2 board, a board with a Xilinx XC7A35T FPGA chip, and was connected to one of the victim machine's PCIe ports as shown in Figure 3.1.

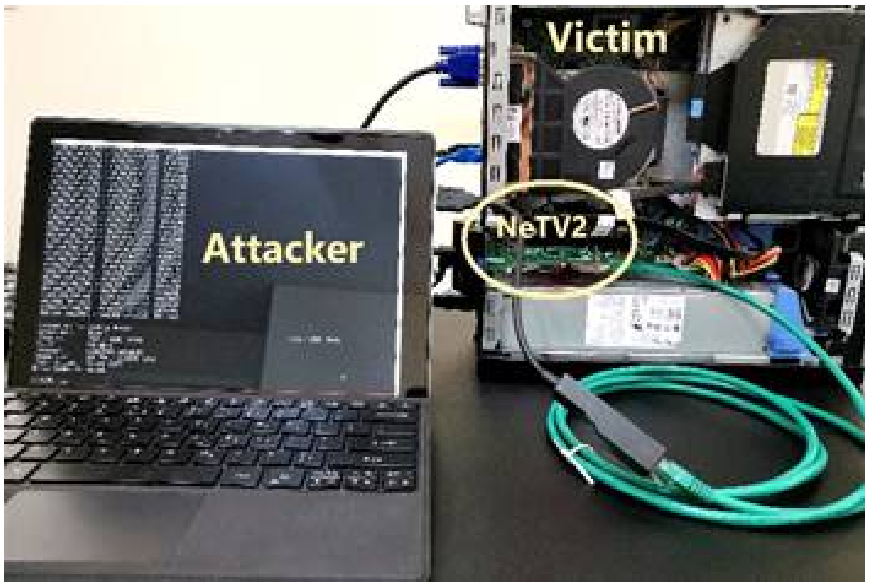
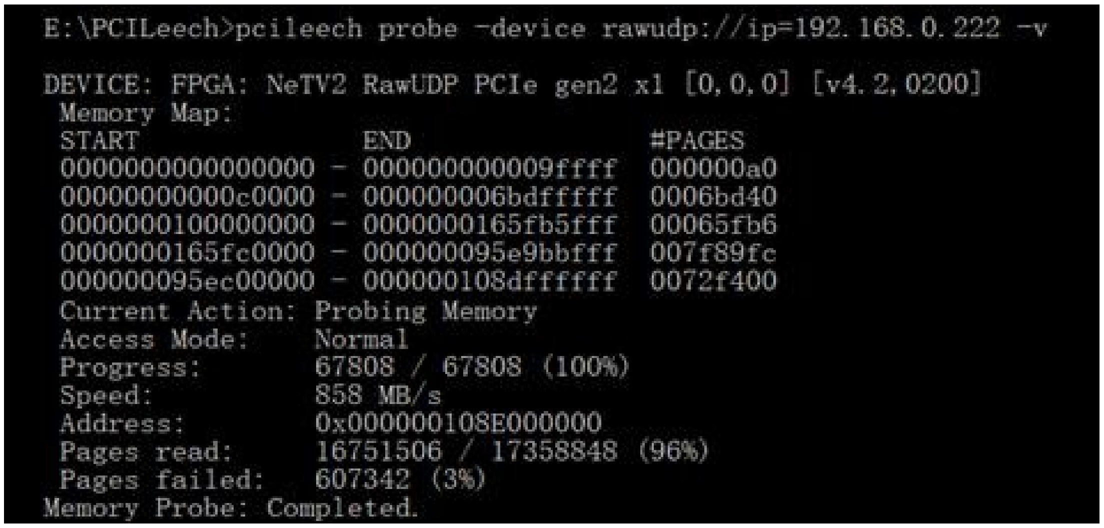


Figure 3.1: Setup for the PCIleech attack.

The Xilinx chip provides a PCIe Bridge, PCIe DMA, as well as soft IP blocks, in addition to being able to access a 64-bit memory space while not needing a kernel module from the victim machine [27]. While the NeTV2 is connected to the victim machine via the PCIe slot, it is connected to the attacker machine through an ethernet cable. The attack does not require any additional software or hardware to be carried out to make the attack undetectable to the victim device. As mentioned earlier in Chapter 2, the attack simply sends a command to the NeTV2 device which will then transfer packets and data to the main memory and back for the attacker to then dump. For the experiment the memory was both probed and dumped, the probe showed the attacker machine that at least 96% of the victim machine memory space was readable as shown in Figure 3.2.

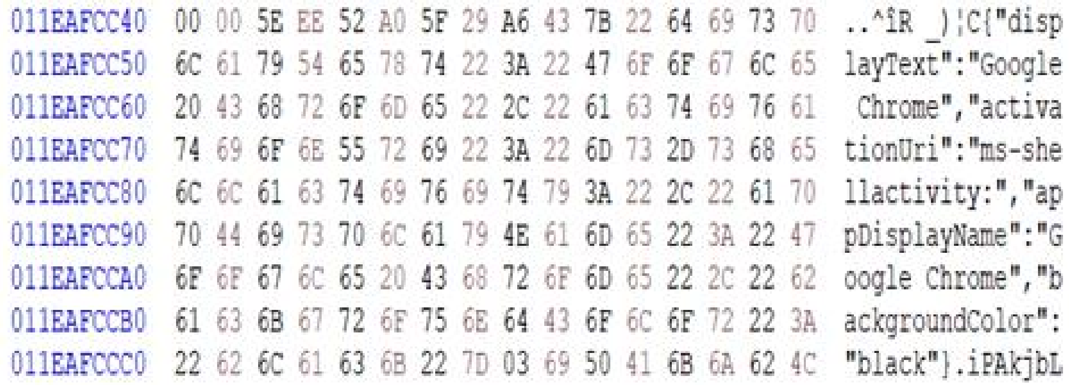


```
E:\PCILeech>pcileech probe -device rawudp://ip=192.168.0.222 -v

DEVICE: FPGA: NeTV2 RawUDP PCIe gen2 x1 [0,0,0] [v4.2,0200]
 Memory Map:
 START                END                  #PAGES
 0000000000000000 - 000000000009ffff  000000a0
 00000000000c0000 - 000000006bdfffff  0006bd40
 0000000100000000 - 0000000165fb5fff  00065fb6
 0000000165fc0000 - 000000095e9bbfff  007f89fc
 000000095ec00000 - 000000108dffffff  0072f400
 Current Action: Probing Memory
 Access Mode:    Normal
 Progress:       67808 / 67808 (100%)
 Speed:          858 MB/s
 Address:        0x000000108E000000
 Pages read:     16751506 / 17358848 (96%)
 Pages failed:   607342 (3%)
Memory Probe: Completed.
```

Figure 3.2: Results from PCIleech memory probe of victim machine.

Along with memory probing, PCIleech is also able to perform memory dumping to the attack as well. Figure 3.3 demonstrates a part of a live memory dump from the victim machine. On the left side of Figure 3.3 is the packet addresses and on the right, Google Chrome is clearly listed. This means that the attacker can see that this piece of memory was used by Google Chrome during the time that the attacker dumped the memory.



```
011EAFCC40  00 00 5E EE 52 A0 5F 29 A6 43 7B 22 64 69 73 70  ..^îR _)¦C{"disp
011EAFCC50  6C 61 79 54 65 78 74 22 3A 22 47 6F 6F 67 6C 65  layText":"Google
011EAFCC60  20 43 68 72 6F 6D 65 22 2C 22 61 63 74 69 76 61   Chrome","activa
011EAFCC70  74 69 6F 6E 55 72 69 22 3A 22 6D 73 2D 73 68 65  tionUri":"ms-she
011EAFCC80  6C 6C 61 63 74 69 76 69 74 79 3A 22 2C 22 61 70  llactivity:","ap
011EAFCC90  70 44 69 73 70 6C 61 79 4E 61 6D 65 22 3A 22 47  pDisplayName":"G
011EAFCCA0  6F 6F 67 6C 65 20 43 68 72 6F 6D 65 22 2C 22 62  oogle Chrome","b
011EAFCCB0  61 63 6B 67 72 6F 75 6E 64 43 6F 6C 6F 72 22 3A  ackgroundColor":
011EAFCCC0  22 62 6C 61 63 6B 22 7D 03 69 50 41 6B 6A 62 4C  "black"}.iPAkjbL
```

Figure 3.3: Excerpt of memory dump obtained from the victim machine.

3.2 Proposed Scheme

This work offers a proposed method of access control by creating a database of trusted devices and creating unique identifiers for each device based on profiling time.

Each device, no matter the manufacturing process, will have variations in current, IR drop, voltage, and unique delays [28] [29]. These unique delays influence device response time, which can be measured and can also create cryptographic functions such as Physical Unclonable Functions (PUFs). Figure 3.4 demonstrates where in the attack the proposed scheme would be used to deny access to the attacker.

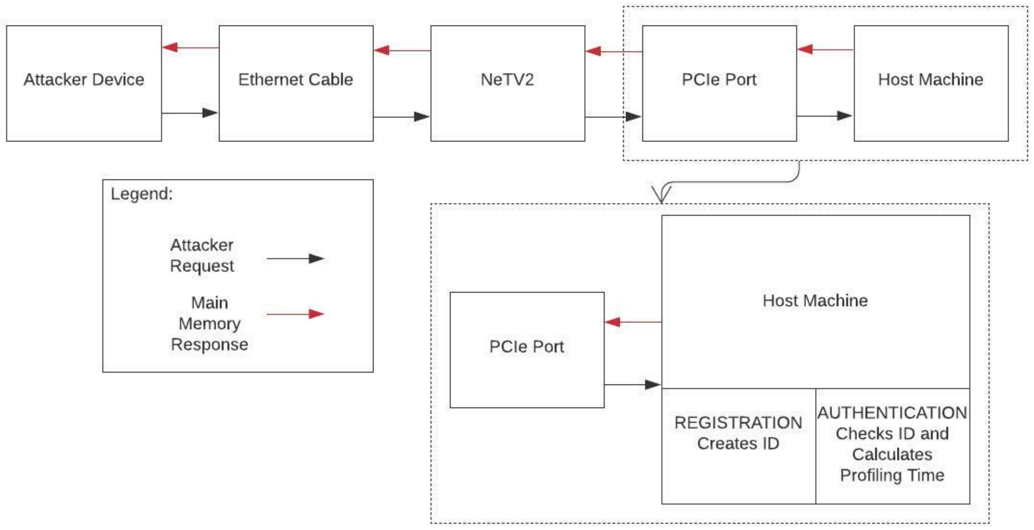


Figure 3.4: Attack flow and proposed scheme.

The proposed method is a two-stage authentication process using a database on the system for PCIe devices before they can use DMA to access the main memory. As shown in Figure 3.5, the first phase is done when the device is connected to the machine for the first time. The system takes sample measurements of the response time delays of the device. By measuring the delays and computing the profiling time for the new device when it is first connected to the system, the profiling time, as well as vendor ID and device ID, of trusted devices is stored in a database on the machine that authenticates devices before giving them access to the main memory. An issue with this approach is that the raw data taken from sample measurements of the delays have high variance to be used for identifying and authenticating devices. To obtain accurate and stable profiling times to make unique identifiers, this paper offers a mathematical process to calculate reliable profiling times from the raw measurements. The second phase of the proposed process is performed during actual run time, taking fewer sample measurements from the device requesting DMA and checking the vendor ID, device ID, and the profiling time of the device with those in the database. If there is a match within the database, the device will be granted access. Both phases involve the construction of an identifier based on profiling time, which is constructed from the Region of Interest (ROI) and Range of Profiling Time (RPT) that are processed from the raw measurement data.

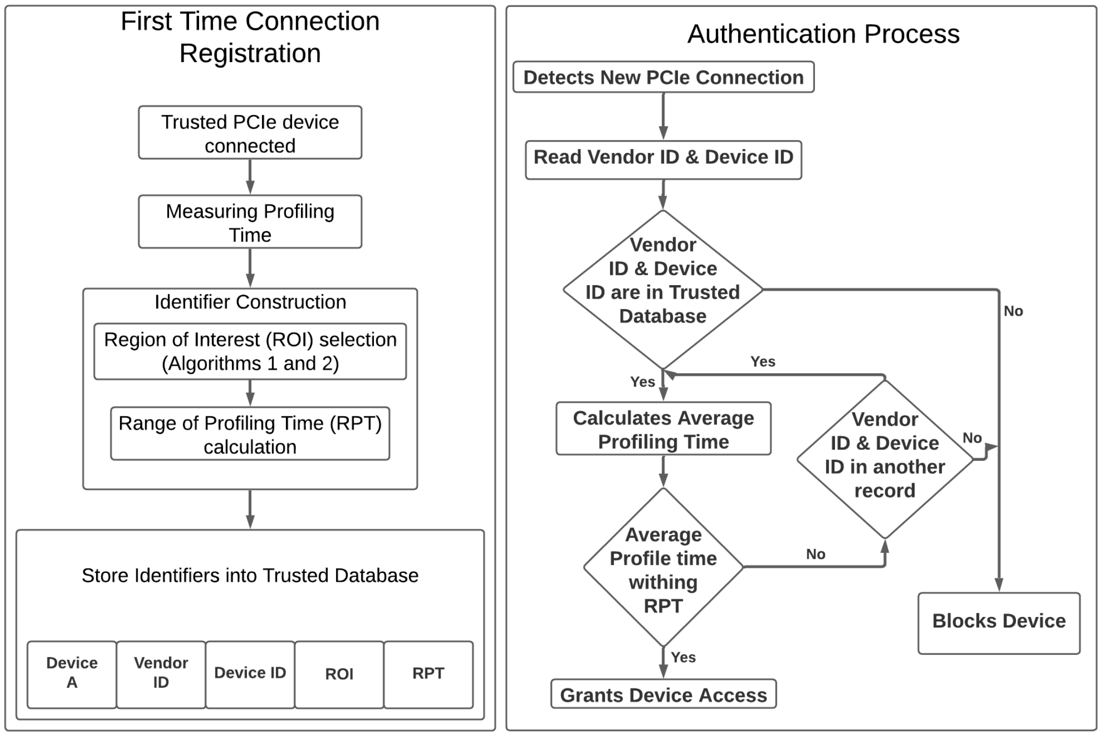

First Time Connection Registration
Trusted PCIe device connected
Measuring Profiling Time
Identifier Construction
Region of Interest (ROI) selection (Algorithms 1 and 2)
Range of Profiling Time (RPT) calculation
Store Identifiers into Trusted Database
Device A
Vendor ID
Device ID
ROI
RPT
Authentication Process
Detects New PCIe Connection
Read Vendor ID & Device ID
Vendor ID & Device ID are in Trusted Database
No
Yes
Yes
Calculates Average Profiling Time
Vendor ID & Device ID in another record
No
Average Profile time withing RPT
No
Blocks Device
Yes
Grants Device Access

Figure 3.5: The registration and authentication process of proposed scheme.

3.2.1 Registration

Having a reliable RPT is paramount for the scheme, in order to have a suitable range and to decrease the amount of noise interfering with the measurements, a broad range of measurements were taken from the devices. For each device, 30 sub-datasets were created from the measurements of response time taken and each of those datasets was made up of 10,000 different response times. The measurements were also organized from the lowest value to the highest for easier processing. Once all the measurements have been taken, the parameters for the data processing is set with R representing how many sub-datasets were created having S by the amount of response time measurements in each sub-dataset [23] [24]. With these parameters the total number of measurements is calculated to be $(R * S)$ for the algorimths, in this case 300,000 measurements in total need to be processed.

The aim of processing the profiling time data is to calculate a ROI, in this case to find a range of measurements that are believed to have the lowest amount of noise possible. As a demonstration of how much variance there is in one dataset, Figure 3.6 is one dataset of measurements taken from device A. It is clear that there is a high amount of variance within the response time measurements. To high an identier be unique and reliable for a specific PCIe device, the proposed scheme makes use of two different algorithms that looks at two different regions [23] [24]. The regions that the algorithms find are base on two different mathematical calculations, one being difference-based and the other being correlation-based.

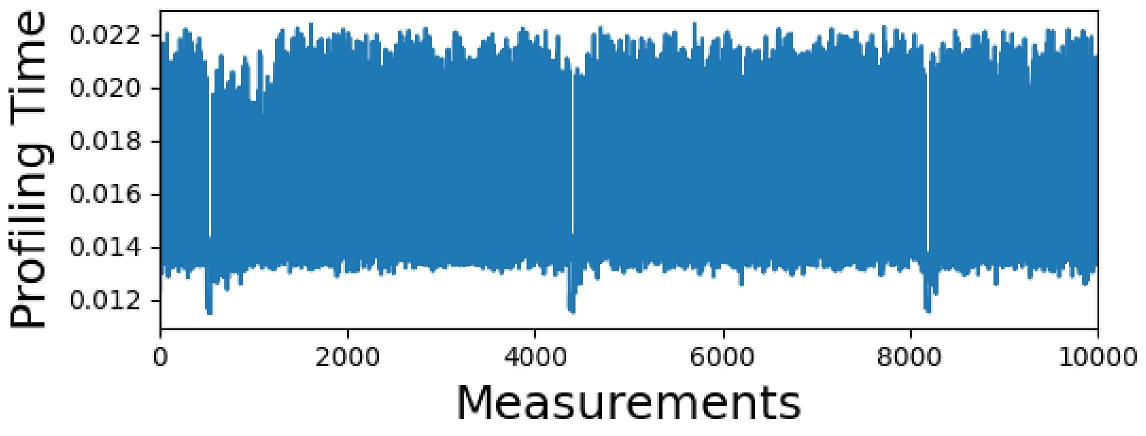


Figure 3.6: An unprocessed measurement from device A.

The Difference-based ROI (DROI) algorithm, denoted as Algorithm 1 [23] [24], takes pairs of different sub-datasets and calculate the difference between them for the same region, starting with the first measurement and ending with the last one. In this algorithm, there are $C(R, 2)$ possible combinations of different pairs for comparison with $(S - L)$ possible regions, where L represents the length of the DROI.

Algorithm 1 DROI Selection

```
    Input: Amount of sub-datasets (R), Amount of measurements in one sub-dataset (S), Length of DROI (L)
    Output: Difference-base ROI (DROI)
  $Res[] \leftarrow$ empty list
  for $i = 1$ to $(S - L)$ do                                  ▷ All possible regions
      for $j = 1$ to $(R - 1)$ do
          for $k = (j + 1)$ to $R$ do                         ▷ All possible combinations
        calculate $AD$          ▷ AD is the absolute difference value between
  sub-dataset$[j][i : i + L]$and sub-dataset$[k][i : i + L]$
        $TD \leftarrow TD + AD$        ▷ TD is the accumulative difference for region i
          end for
      end for$Res[i] \leftarrow TD$ $TD \leftarrow 0$
  end forDROI $\leftarrow [d : d + L]$          ▷ d is the index of the minimum value Res[]
```

With the Correlation-based ROI (CROI) algorithm, denoted as Algorithm 2 [23] [24], a correlation coefficient, a statistical index for the dependency of variables, is computed for pairs of sub-datasets. Specifically, the Pearson's correlation coefficient, which is a measurement of linear correlation, of all the regions in the pairs is computed.

Algorithm 2 CROI Selection

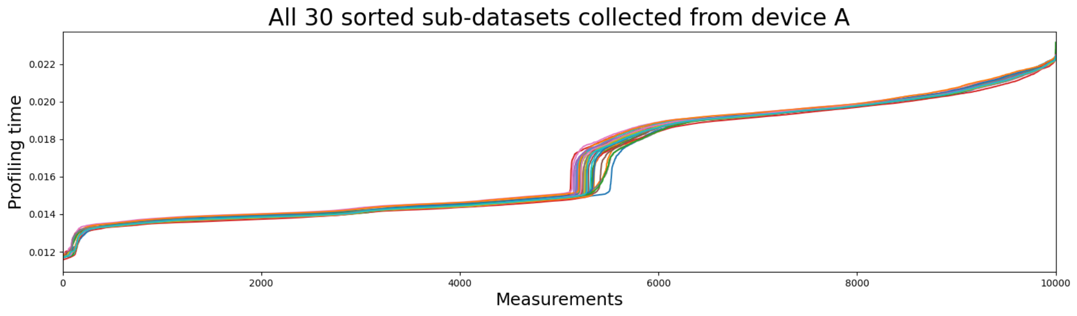
```
    Input: Amount of sub-datasets (R), Amount of measurements in one sub-dataset (S), Length of CROI (L)
    Output: Correlation-base ROI (CROI)
  $Res[] \leftarrow$ empty list
  for $i = 1$ to $(S - L)$ do                                  ▷ All possible regions
      for $j = 1$ to $(R - 1)$ do
          for $k = (j + 1)$ to $R$ do                         ▷ All possible combinations
        calculate $CC$          ▷ CC is the correlation coefficient between
  sub-dataset$[j][i : i + L]$and sub-dataset$[k][i : i + L]$
        $TC \leftarrow TC + CC$        ▷ TC is the accumulative coefficient for region i
          end for
      end for$Res[i] \leftarrow TC$ $TC \leftarrow 0$
  end forCROI $\leftarrow [d : d + L]$          ▷ d is the index of the minimum value Res[]
```

Both algorithms use Res to represent a list generated from the data processed from the measurements, for Algorithm 1 it contains a list of differences of all points from the sub-dataset pairs and for Algorithm 2 contains the correlations of different points from the sub-data sets. Both use L to represent the length of their region, more specifically $[i : i + L]$, starting from the $i^t h$ point [23] [24]. For the DROI Res, only the lowest differences are used, due to the fact that the lower the difference is, the more similar the points are, which in turn means a lesser amount of noise. For the CROI Res, the higher the values are the better, as a high correlation coefficient means higher similarity and in turn means less noise from the measurements. Once the ranges are calculated from the algorithms, they are used on to a graph of all 30 sub-datasets, sorted, and where the DROI and the CROI intersect with each other, that is considered to be the ROI and the region that is used for the unique identifier.

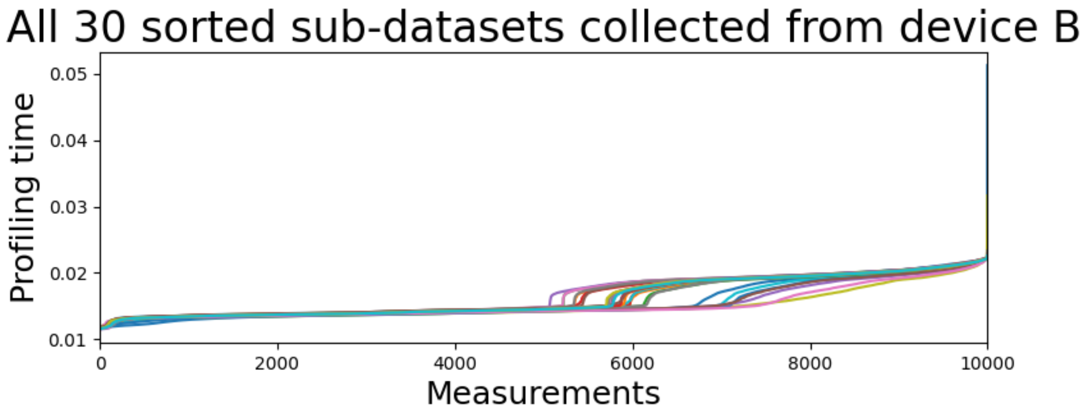
For testing the proposed access control scheme, a number of TL-WN881ND wireless PCIe adapters from TP-LINK were used to get profiling time measurements for both phases of the scheme [30]. Since the devices are the same model and from the same manufacturer, they will most likely have the same Vendor ID and Device ID, meaning that the to tell the apart it will have to be done by measuring the difference in response times and use it for the profiling times. The delay that was measured for the profiling time was the time it took to read the configuration space on the PCIe device, which was used for constructing the identifier.

As shown from Figure 3.6, the variance of one sample measurement from one device can vary drastically, specifically in device A, the variance is between 11 ms to 23 ms in response time, from 10,000 samples measured. While the range is fairly consistent across the 10,000 samples, it could cause aliasing if the other devices have the same behavior. In order to less the chance of aliasing, the measurements are sorted from lowest response time, to highest response time.

Not only can the manufacturing process cause the high variations seen, the environment, such a temperature and other sources of variation, can also contribute to the noise. 30 sub-datasets were taken in order to try to mitigate the effect that environmental variations have on the data processing. Figure 3.7 shows 30 sub-datasets graphed from two devices used in the experiment, which will be known as device A and device B respectively. From the visuals of the graphs, there is clearly high variance for both devices, causing a wide range of measurements to be taken and recorded. If the average profiling times are taken from the wrong region, then the identifier could be vulnerable to aliasing and bit flips.



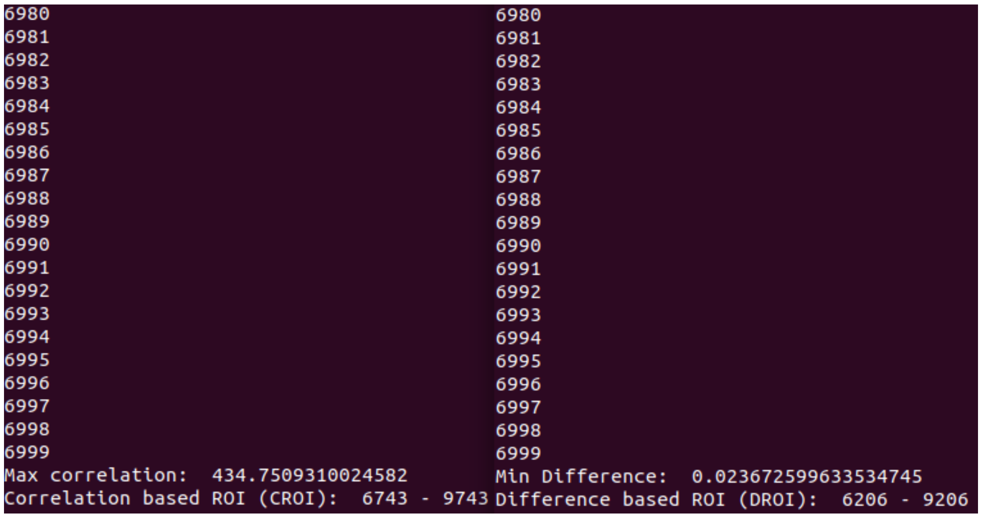
(a) Graph of the 30 sub datasets of measurements taken from device A



(b) Graph of the 30 sub datasets of measurements taken from device B

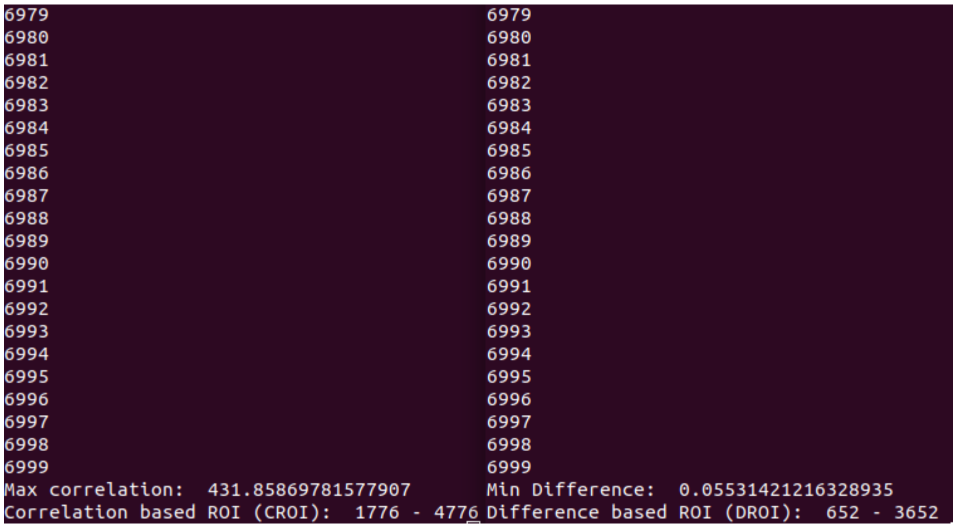
Figure 3.7: Measurements graphed from device A (a) and device B (b).

During test configuration, the algorithms for the DROI and the CROI both used 3000 as their length parameter. Figure 3.8 shows the results of Algorithm 1 and Algorithm 2 when used to process the data collected from device A and device B.



```
6980                                         6980
6981                                         6981
6982                                         6982
6983                                         6983
6984                                         6984
6985                                         6985
6986                                         6986
6987                                         6987
6988                                         6988
6989                                         6989
6990                                         6990
6991                                         6991
6992                                         6992
6993                                         6993
6994                                         6994
6995                                         6995
6996                                         6996
6997                                         6997
6998                                         6998
6999                                         6999
Max correlation:  434.7509310024582          Min Difference:  0.023672599633534745
Correlation based ROI (CROI):  6743 - 9743   Difference based ROI (DROI):  6206 - 9206
```

(a) The calculated DROI and CROI of device A



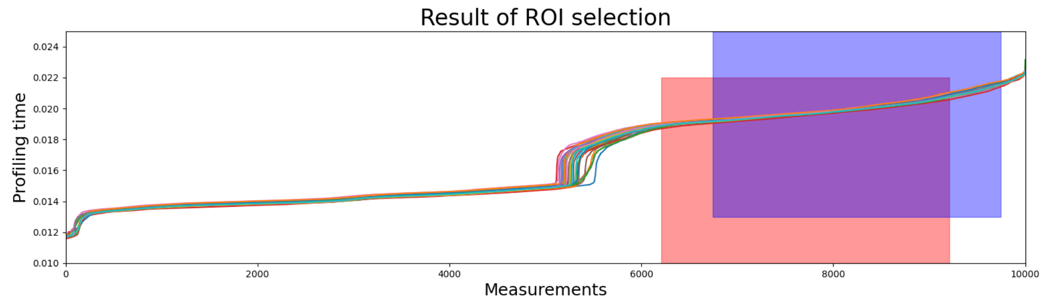
```
6979                                         6979
6980                                         6980
6981                                         6981
6982                                         6982
6983                                         6983
6984                                         6984
6985                                         6985
6986                                         6986
6987                                         6987
6988                                         6988
6989                                         6989
6990                                         6990
6991                                         6991
6992                                         6992
6993                                         6993
6994                                         6994
6995                                         6995
6996                                         6996
6997                                         6997
6998                                         6998
6999                                         6999
Max correlation:  431.85869781577907         Min Difference:  0.05531421216328935
Correlation based ROI (CROI):  1776 - 4776   Difference based ROI (DROI):  652 - 3652
```

(b) The calculated DROI and CROI of device B

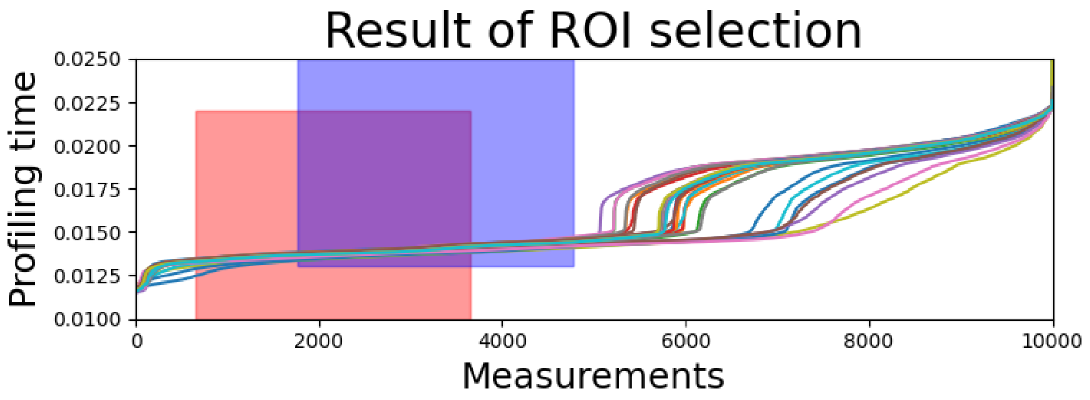
Figure 3.8: The DROI and CROI of device A (a) and device B (b) after measurement data has been processed.

From the results of Figure 3.8, the DROI of device A is found to be within the 6206th point and the 9206th point, while device B seems to have its DROI between the 652nd point and the 3652nd point. For the CROI, the ideal region for Device A would be between the 6743rd point and the 9743rd point with the CROI for device B being the 1776th point and the 4776th point. Already showing that the identifiers for devices A and B can be vastly different from each other. The lengths of the CROI and

DROI can be changeable though for the purposes of this experiment they were set to 3000. This was done so that the device could have a sizable ROI because if it was too narrow then the rate of authentication could be reduced and a trusted device could be denied access. In contrast, if the length is too large and the profiling times derived from the ROI could possibly produce non-unique identifiers. From these results, the calculated regions are used to find where measurements are in both regions at the same time, as shown in Figure 3.9



(a) Device A datasets with DROI and CROI



(b) Device B datasets with DROI and CROI

Figure 3.9: The DROI (red) and the CROI (blue) calculated from the measurements taken from device A (a) and device B (b).

From Figure 3.9 it is clear that for both devices, the highlighted section of DROI and CROI and the overlap. The DROI is made of the lowest differences and the CROI is made from the highest correlations from the sub-datasets. The region where the

DROI and the CROI intersect therefore contains both the lowest difference and the highest correlations, meaning that it is the region with possibly the lowest noise with the highest reliability over the different points of the sub-data sets. In this case, the ROI for device A was found to be between the 6743rd point and the 9206th point. As for device B, the ROI was calculated to be the area ranging from the 1776th point to the 3652nd point.

Using the ROI that was calculated, the database can use it to identify the RPT for each device and create a unique identifier. This was done by taking the ROI found for both devices and looking at all the points in the 30 sub-datasets that were in that region. All the measurements that fell in the ROI were average for each different sub-dataset. Those averages were used to construct a RPT in order to be the unique identifier. Figure 3.10 shows that the RPT of device A extends from 19718.66 µs to 19977.18 µs and that the RPT of device B is across 13330.72 µs to 14249.39 µs. As there is no overlap between these two devices that were registered, it is possible to create a unique identifier for both of them in the systems database. The Vendor ID, Device ID, RPT, and ROI for each device would be stored onto the host machines trusted database and used to authenticate devices when they ask for access.

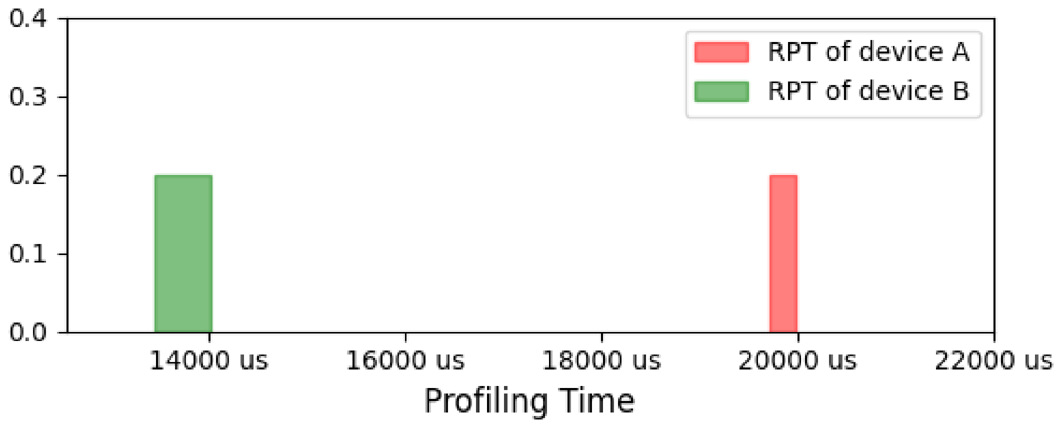


Figure 3.10: Comparison of profiling times of device A and device B.

3.2.2 Authentication

To test the authentication phase, three devices were used to gather response time data. The devices were device A, device B, and device C respectively. For this process, 10,000 response time measurements were taken from all three devices separately, 30 times to create another 30 sub-datasets. In this process, only the DROI and the CROI of device A was calculated in order to find the ROI. From the data processing the CROI was found to be a range of the 6991st point to the 9991st point and the DROI was calculated to be frome the 5464th point and the 8464th point. From this data, the ranges overlap from the 6991st point to the 8464th point, which will be used as the ROI in this case.

Once the ROI was calculated for device A, it was used to create the RPT for all the devices. Figure 3.11 shows the results of the RPT calculations for each device. Even though the devices used the same ROI, the RPT still came out to be completely different with no overlap between the devices.

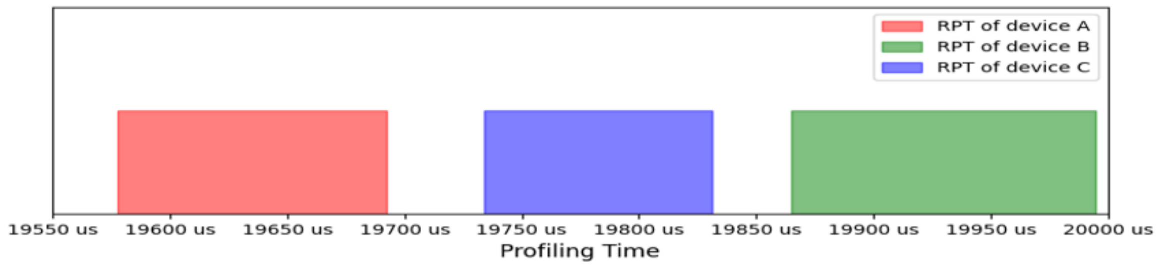


Figure 3.11: The RPTs from each device using the same ROI.

This highlights the variances between devices that are the same model and from the same manufacturer. The RPT for device A was found to be 19577.90 µs to 19692.26 µs, device B's RPT was calculated to be 19865.06 µs to 19994.50 µs, and device C was found to have a RPT of 19734.17 µs to 19831.31 µs.

Using the RPT calculated, the devices response times were measured 10,000 times again. The ROI of device A was used yet again to find the average profiling time for each measurement. For this experiment, a range of profiling times was not calculated, rather the average time found and compared to the RPT found for each device. This measurement and calculation was done 20 times for each device and each device was able to get averages within the RPT for their respective devices 100% of the time [23].

Both the registration phase of the experiment and the authentication phase were timed, specifically the amount of time it took to measure the device response times and how long it took to construct the unique identifier. The times were recorded in Table 3.1.

Table 3.1: Time to complete each phase.

Phase	Time of Measurement	Time of Construction
Registration	100 minutes	3.4 minutes
Authentication	15.7 minutes	$4 * 10^{-7}$ minutes

For the registration phase it took 100 minutes to measure all the response times needed for the profiling time averages and took 15.7 minutes to construct the unique identifier for a device. The authentication phase took 3.4 minutes to measure the profiling time samples and took $4*10^{-7}$ minutes to construct the identifier to compare to the database for the device. The large time discrepancy between the registration phase and the authentication phase comes from the fact that the registration phase needs a larger amount of response times to create a RPT. Authentication does not need to construct a range, so the time was based off how long it took to create an average profiling time from 1 sub-data set. The timing can also vary depending on the CPU and how efficiently it can perform the necessary calculations. For this experiment a Core i5-3470 was used for each measuring and identifier construction.

CHAPTER 4: CONCLUSIONS

In this thesis, we have demonstrated the effectiveness of a DMA attack and proposed an ROI based access control method for PCIe devices used in DMA attacks. The DMA attack has shown that this method of side-channel analysis can be used to violate confidentiality. With this method, attackers can have full access to a user's memory and even perform live illegal memory dumps to extract secret information.

The proposed process used unique delay response times to create identifiers for the devices that cannot be replicated due to environmental variances during manufacturing. The method has also shown that even with similar devices of the same manufacture and model, there is no overlap between the profiling time of the devices. The scheme does not require modification to the hardware of the machine nor to the protocols and was designed in mind to be able to test and check a wide range of devices, even some that are not programmable. The main idea behind the design was for it to be lightweight and could be used as a general access control method for a wide range of PCIe devices. There has also been little to no impact on the performance of the host machine, a table that is constructed to contain the information of 30 devices was found to only require 1.81 KB of space [23] [24].

CHAPTER 5: FUTURE WORK

One of the main issues with the proposed scheme is the time overhead due to the registration process. One possible way to reduce the measurement and construction time is using machine learning to authenticate the devices [23][24]. Other lines of research would be to modify the DMA process. One such are to look at in the PCIe architecture would be the root complex, which is the endpoint that connects the PCIe device to the RAM and the CPU [31].

To improve the hit rate, more measurements need to be taken during the authentication phase. The same devices were measure on the same machine, yet the hit rate was not adequate enough for the pillar of availability. Using a large range of devices would also help with analyzing the effectiveness of the scheme.

Another way to improve upon the proposed scheme would be to have the PCIe device be able to calculate its own RPT and send it to the database to check for its identifiers. This approach could help negate the latency in the PCIe bus to allow for more accurate readings to create identifiers. There are also devices and software that are designed to measure and analyze PCIe devices more accurately, but for the purpose of this experiment, it was to create a general access control scheme that could be used on a wide range of host machines and register a wide range of PCIe devices.

REFERENCES

[1] PurpleSec, "2021 cyber security statistics trends & data." 2021. Available at `https://purplesec.us/resources/cyber-security-statistics/#Vulnerabilities`.

[2] G. E. Suh, D. Clarke, B. Gassend, M. van Dijk, and S. Devadas, "Aegis: Architecture for tamper-evident and tamper-resistant processing," in *ACM International Conference on Supercomputing 25th Anniversary Volume*, (New York, NY, USA), pp. 357–368, Association for Computing Machinery, 2003.

[3] F. McKeen, I. Alexandrovich, A. Berenzon, C. V. Rozas, H. Shafi, V. Shanbhogue, and U. R. Savagaonkar, "Innovative instructions and software model for isolated execution.," *Hasp@ isca*, vol. 10, no. 1, 2013.

[4] Y. Xu, W. Cui, and M. Peinado, "Controlled-channel attacks: Deterministic side channels for untrusted operating systems," in *2015 IEEE Symposium on Security and Privacy*, pp. 640–656, 2015.

[5] Microsoft, "Bitlocker countermeasures (windows 10) - windows security." 2019. Available at `https://docs.microsoft.com/en-us/windows/security/information-protection/bitlocker/bitlocker-countermeasures#pre-boot-authentication`.

[6] D. Abramson, J. Jackson, S. Muthrasanallur, G. Neiger, G. Regnier, R. Sankaran, I. Schoinas, R. Uhlig, B. Vembu, and J. Wiegert, "Intel virtualization technology for directed i/o.," *Intel technology journal*, vol. 10, no. 3, 2006.

[7] Q. Han, L. Wu, and X. Zhang, "Research on side channel attack for usb key," in *2016 12th International Conference on Computational Intelligence and Security (CIS)*, pp. 594–597, IEEE, 2016.

[8] K.-A. Shim, "A survey of public-key cryptographic primitives in wireless sensor networks," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 577–601, 2015.

[9] B. Ruytenberg, "When lightning strikes thrice: Breaking thunderbolt 3 security." 2020. Available at `https://thunderspy.io/`.

[10] B. Morgan, E. Alata, V. Nicomette, and M. Kaâniche, "Bypassing iommu protection against i/o attacks," in *2016 Seventh Latin-American Symposium on Dependable Computing (LADC)*, pp. 145–150, IEEE, 2016.

[11] V. E. in Informatics, "Des challenge." Available at `http://cs-exhibitions.uni-klu.ac.at/index.php?id=263`.

[12] G. K.-H. Jonathan Corbet, Alessandro Rubini, "Chapter 15. memory mapping and dma." 2005. Available at `https://www.oreilly.com/library/view/linux-device-drivers/0596005903/ch15.html`.

[13] Intel, "8237a high performance programmable dma controller." 2021. Available at `https://www.lo-tech.co.uk/downloads/manuals/intel/Intel-8237A-datasheet.pdf`.

[14] S. V. Savage and J. M. Harris, "Direct memory access controller for improved system security, memory to memory transfers, and interrupt processing," Jan. 10 1989. US Patent 4,797,853.

[15] F. O. C. Solutions, "How to distinguish pci vs pci express?." 2020. Available at `ttps://www.cables-solutions.com/pci-vs-pci-express.`tml.

[16] D. J. Miller, P. M. Watts, and A. W. Moore, "Motivating future interconnects: A differential measurement analysis of pci latency," in *Proceedings of the 5th ACM/IEEE Symposium on Architectures for Networking and Communications Systems*, ANCS '09, (New York, NY, USA), pp. 94–103, Association for Computing Machinery, 2009.

[17] D. D. Sharma, "A low latency approach to delivering alternate protocols with coherency and memory semantics using pci expressÂ® 6.0 phy at 64.0 gt/s," in *2021 IEEE Symposium on High-Performance Interconnects (HOTI)*, pp. 35–42, 2021.

[18] R. Neugebauer, G. Antichi, J. F. Zazo, Y. Audzevich, S. López-Buedo, and A. W. Moore, "Understanding pcie performance for end host networking," in *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, SIGCOMM '18, (New York, NY, USA), pp. 327–341, Association for Computing Machinery, 2018.

[19] B. Böck and S. B. Austria, "Firewire-based physical security attacks on windows 7, efs and bitlocker," *Secure Business Austria Research Lab*, 2009.

[20] U. Frisk, "Direct memory attack the kernel." 2016. Available at `https://media.defcon.org/DEF%20CON%2024/DEF%20CON%2024%20presentations/DEF%20CON%2024%20-%20Ulf-Frisk-Direct-Memory-Attack-the-Kernel.pdf`.

[21] U. Frisk, "Pcileech." 2016. Available at `https://github.com/ufrisk/pcileech`.

[22] T. M. John, S. K. Haider, H. Omar, and M. Van Dijk, "Connecting the dots: Privacy leakage via write-access patterns to the main memory," *IEEE Transactions on Dependable and Secure Computing*, vol. 17, no. 2, pp. 436–442, 2017.

[23] Y. Gui, A. S. Siddiqui, G. S. Nicholas, M. Hughes, and F. Saqib, "A lightweight delay-based authentication scheme for dma attack mitigation," in *2021 22nd International Symposium on Quality Electronic Design (ISQED)*, pp. 263–268, IEEE, 2021.

[24] Y. Gui, C. Bhure, M. Hughes, and F. Saqib, "A delay-based machine learning model for dma attack mitigation," *Cryptography*, vol. 5, no. 3, 2021.

[25] AMD, “Amd i/o virtualization technology (iommu) specification.” 2021. Available at `https://www.amd.com/system/files/TechDocs/48882_IOMMU.pdf`.

[26] D. Wendlandt, D. G. Andersen, and A. Perrig, “Perspectives: Improving ssh-style host authentication with multi-path probing.,” in *USENIX Annual Technical Conference*, vol. 8, pp. 321–334, 2008.

[27] Xilinx, “Pci express and xilinx technology.” 2020. Available at `https://www.xilinx.com/products/technology/pci-express.html`.

[28] G. E. Suh and S. Devadas, “Physical unclonable functions for device authentication and secret key generation,” in *2007 44th ACM/IEEE Design Automation Conference*, pp. 9–14, IEEE, 2007.

[29] J. W. Lee, D. Lim, B. Gassend, G. E. Suh, M. Van Dijk, and S. Devadas, “A technique to build a secret key in integrated circuits for identification and authentication applications,” in *2004 Symposium on VLSI Circuits. Digest of Technical Papers (IEEE Cat. No. 04CH37525)*, pp. 176–179, IEEE, 2004.

[30] TP-LINK, “Ti-wn881nd.” 2020. Available at `https://www.tp-link.com/il/home-networking/adapter/tl-wn881nd/`.

[31] P. K. Tanwar, O. P. Thakur, K. Bhimani, G. Purohit, V. Kumar, S. Singh, K. S. Raju, I. Ishii, and S. Raut, “Zynq soc based high speed data transfer using pcie: A device driver based approach,” in *2017 14th IEEE India Council International Conference (INDICON)*, pp. 1–6, IEEE, 2017.