5-2022

Hardware Isolation Approach to Securely Use Untrusted GPUS in Cloud Environments for Machine Learning

Lucas D. Hall
The University of Texas Rio Grande Valley

HARDWARE ISOLATION APPROACH TO SECURELY USING UNTRUSTED GPUS IN CLOUD ENVIRONMENTS FOR MACHINE LEARNING

A Thesis

by

LUCAS D. HALL

Submitted in Partial Fulfillment of the

Requirements for the Degree of

MASTER OF SCIENCE

Major Subject: Computer Science

The University of Texas Rio Grande Valley

May 2022

HARDWARE ISOLATION APPROACH TO SECURELY USING UNTRUSTED GPUS IN CLOUD ENVIRONMENTS FOR MACHINE LEARNING

A Thesis
by
LUCAS D. HALL

COMMITTEE MEMBERS

Dr. Lei Xu
Chair of Committee

Dr. Sheikh Ariful Islam
Committee Member

Dr. Honglu Jiang
Committee Member

May 2022

ABSTRACT

Hall, Lucas D., Hardware Isolation Approach to Securely using Untrusted GPUs in Cloud Environments for Machine Learning. Master of Science (MS), May, 2022, 37 pp., 3 tables, 4 figures, 15 references.

Machine Learning (ML) is now a primary method for getting useful information out of the immense volumes of data being generated and stored in society today. Useful data is a commodity for training ML models and those that need data for training are often not the owners of the data leading to a desire to use cloud-based services. Deep learning algorithms are best suited to run on a graphical processing unit (GPU) which presents a specific problem since the GPU is not a secure or trusted piece of hardware in the cloud computing environment.

In this paper, we will analyze some current methods of performing ML in the cloud using untrusted hardware and propose FIGHTE: full isolation of GPU hardware for trusted execution, a new hardware implementation capable of physical isolation. FIGHTE should allow for securely using a GPU for ML in the cloud even for various parties involved.

TABLE OF CONTENTS

Page

ABSTRACT iii
TABLE OF CONTENTS iv
LIST OF TABLES vi
LIST OF FIGURES vii
CHAPTER I. INTRODUCTION 1
1.1 Background Information 1
1.2 Problem Overview 2
CHAPTER II. SOFTWARE SOLUTIONS 4
2.1 Homomorphic Encryption Method 4
2.2 Verification Methods 5
2.3 Multi-party Computation 6
2.4 Limitations of Software-based Secure Machine Learning in the Cloud 7
CHAPTER III. HARDWARE SOLUTIONS 9
3.1 Pure TEE Method 9
3.2 Isolation by Modified Interconnect 12
3.3 Emulated GPU TEE 12
3.4 NVIDIA H100 PCIe Gen 5 GPU 13
3.4.1 Other H100 Devices 14
3.5 Limitations on Hardware-based Secure ML in the Cloud 14
CHAPTER IV. FIGHTE: FULL ISOLATION OF GPU HARDWARE FOR TRUSTED EXECUTION 16
4.1 Introduction 16
4.2 Requirements 17
4.2.1 Basic Requirements 17
4.2.2 Attestation 18
4.2.3 Modification and Design Requirements 18
4.2.4 Interruptibility Requirements 19

4.3 Isolation Concept . . . 19
4.3.1 GPU Modifications . . . 19
4.4 Data Flow Through Hardware . . . 22
4.5 Software Implementation . . . 24
4.5.1 Primary Software . . . 24
4.5.2 Secondary Software . . . 26
4.6 Implementations and Testing . . . 26
4.7 Security Implications . . . 28
4.8 Performance Implications . . . 29
4.9 Other Challenges and Limitations . . . 30
4.10 Possible Drawbacks of this Method . . . 31
4.11 Comparison with NVIDIA H100 . . . 31
CHAPTER V. CONCLUSION AND FUTURE RESEARCH DIRECTIONS . . . 33
5.1 Conclusion . . . 33
5.2 Future Research . . . 33
REFERENCES . . . 35
BIOGRAPHICAL SKETCH . . . 37

LIST OF TABLES

Page

Table 4.1: List of Security Requirements 17
Table 4.2: List of Devices for GPU card 20
Table 4.3: Comparison of Various Common Speeds 24

LIST OF FIGURES

Page

Figure 4.1: Concept View of Modified GPU Card to Act as a Standalone Subsystem 20
Figure 4.2: Flow of Data through the Hardware 23
Figure 4.3: Workflow of Primary Program. 25
Figure 4.4: Workflow of Secondary Program 27

CHAPTER I

INTRODUCTION

1.1 Background Information

Machine learning has become an integral part of our society. It is used in phones, cars, drones, robots, and even automated helicopters as well as all sorts of other software applications. At this time ML is really the driving force behind what is commonly considered Artificial Intelligence (AI). This has been as much a boon for the tech industry as it has been a liability. Applications and uses based on ML are regularly the subject of news reports typically covering an issue where an AI program did something unexpected such as a wrecked self-driving car or wrongfully identified a person. There have been notable instances in recent news where ML based applications have created undesired and unexpected outcomes due either to insufficient training data and/or human error.

What has been missing from reports are incidents of ML being compromised which might be misleading in that a lack of known security breaches constitutes high security amongst ML. The reality is that security in ML is quite different than other forms of software security. Certainly, many aspects are the similar however the nature of ML presents a unique problem set regarding security. Whereas there are many types of ML, the focus here is on deep learning algorithms which may take copious amounts of time and resources to complete. Due to amount of time and data required to complete the model training process, adversarial attacks may take place throughout the training process affecting either the dataset or the model depending on the nature of the attack. The time involved in training also leads to using GPUs for training models which can lower training times drastically compared to central processing units (CPU) but can also exacerbate security issues as

GPUs do not have the security technology present on many CPUs used in cloud computing. The attacks occurring during this process generally are done to either steal the dataset, steal the model, or to poison the model such that once used the results are modified from what is to be expected or greatly reduce the accuracy.

Xue et al. 2020 provides an in depth look at ML security issues and some of the defenses against them currently available. Many of the security concerns described are the same that this paper attempts to solve in the presented solution, though his paper is limited in scope to longer deep neural network (DNN) processes. The longer processes are vulnerable to specific attacks such as training data poisoning, data set theft, model theft, and others. The other types of ML may still benefit from the proposed solution. Many of the defenses described relate to a single type of attack or simply maintaining privacy of a dataset either though encryption, verification, or detection and incur considerable overhead which may be costly or unmanageable in practice.

1.2 Problem Overview

The problem with security in the cloud for ML is both multifaceted and unique. Because the process can take considerable amount of processing resources and requires very large amounts of data, the process of training a model takes considerable amount of time. As the need for more accurate models grows, the time needed to train such models grows as well. Furthermore, the training process is not best performed on CPUs. ML uses a lot of matrix operations that GPUs are far more efficient at processing than general purpose CPUs; however, GPUs do not have any advanced security features that are common to modern CPUs.

What makes ML in the cloud a unique problem is that the training of a model takes a lot of time, days in some cases. This creates a large window of opportunity for an attacker. Given days' worth of time while a process is running, it would appear to very realistic for an attacker or even multiple attackers to be to analyze the process and launch successful attacks. The most secure way currently to run the process is using the CPU and leveraging the various security features of the CPU. This is of course the slowest way and is still prone to security breaches. In order to greatly improve runtimes, it seems only natural to use the GPU which leads to inevitable tradeoffs between

performance and security leaving no truly secure or full speed method.

Other problematic aspects must also be addressed. The dataset and model must be kept secure and confidential at all times which generally means encrypted. Yet during training and testing phases these items need to be at least in part decrypted using most methods leaving data and model vulnerable particularly if using a GPU. The datasets can be very large, at times possibly in the terabyte range, creating storage and other hardware considerations while developing a solution.

In summary, GPUs hold the key to efficient ML training in the cloud yet remain an untrusted piece of hardware. Any solution developed to utilize the GPU in this environment must take into consideration aspects of ML such as the ability to keep data encrypted and inaccessible while not encrypted as well as to ensure the hardware has the capacity to house and work with exceedingly large datasets.

CHAPTER II

SOFTWARE SOLUTIONS

As with many security issues, it is fast and cost efficient to implement software-based security solutions ahead of seeking hardware-based solutions. The goal of using an untrusted GPU for secure ML training is certainly no exception. A plethora of possible solutions have arisen. To understand the drawbacks of many of the proposed solutions, a non-comprehensive review is provided for perspective. These solutions include variations of homomorphic encryption schemes, verification schemes, and other encryption schemes such as multi-party computation. Much of the works are done using trusted execution environments (TEEs) in some regards.

2.1 Homomorphic Encryption Method

Homomorphic encryption is a method that allows for data to be used while remaining in an encrypted state. In terms of ML, this means being able to keep a dataset encrypted in such a way that the model can still be trained and tested on the data without first being decrypted. Immediately knowing that all data stays in an encrypted state ensures a high level of privacy and it is doubtful that any additional overhead would be incurred due to using the data in such a state.

Wu, Du, and Yuan 2020 have demonstrated a technique for encrypting data, images in this study, in such a way that the features are preserved well enough that the images can be used to train a model with high accuracy. The encryption is strong enough that any adversary would need direct knowledge of the encryption scheme in order to gain access to protected data. Any leaked data would still be in an encrypted state and thus highly unusable outside the intended scope. Despite that this method was not intended to be used specifically for the problem at hand, the method seems very close to a standalone solution. The data is protected by encryption and the model is useless

except without the owner having the encryption key. It would need protection from various other attacks such as poisoning attacks or bomb attacks.

One drawback may be that the model only works for the encrypted data and not the original data, for testing and application. It is stated that all usage of the model will need to be done using data encrypted in the same manner as the data the model was trained and tested with. This means that the owner of the model will have to encrypt any data to be worked on using the encryption method and key for that model. With that in mind, it would seem that the encryption method could become widely known very quickly and could lead to theft of data if the encryption scheme is not strong enough or keys are leaked. Still the concept is pretty solid and could likely be improved upon. It may prove useful in particular if combined with other security solutions.

2.2 Verification Methods

Several approaches have been made to securely use the GPU for ML by means of verifying processes. The methods typically rely in part on a CPU provided TEE along with software to delegate certain aspects of training to an untrusted GPU. The verification aspect serves to ensure a high degree of integrity in the operations that are being outsourced to the GPU. Indeed, the main benefit of these methods is to increase performance of the process while maintaining integrity foremost to ensure that the model is trained accurately.

Tramèr and Boneh 2018 propose Slalom which is a method that outsources the costly linear computations of the ML process to the GPU and is primarily focused on the testing phase of the ML process. The authors used Intel SGX as a source of TEE and a method for verifying integrity based on Freivalds' Algorithm. Freivalds' Algorithm is a probabilistic method to determine whether one matrix is the product of two matrices and has a time complexity of $O(n^2)$ whereas regular matrix multiplication is $O(n^3)$. It is a considerably faster method than multiplying 2 matrices together and allows the ML model to know that the output is correct and that no changes have occurred during the process using the untrusted GPU. Overall, Slalom proves to be 4 to 20 times faster than pure TEE methods though it still incurs quite a bit of overhead and appears to require large amounts of RAM or access to fast storage devices. Combined with privacy protections, it does provide a

reasonable number of protections for its purpose.

Where the focus in Slalom is the testing phase, GOAT, created by Asvadishirehjini, Kantarcioglu, and Malin 2020, is a method that uses random verifications to ensure integrity in the training phase of the ML process. This serves to detect specific attacks that would occur during the training phase such as a poisoning attack. In this method, Intel SGX is once again leveraged as a basis for privacy while outsourcing the dataset in plaintext to the GPU. Privacy is not a major concern in this approach where the focus is specifically integrity which leaves the TEE usage limited to administrative tasks in operating GOAT. This method works by sending mini batches of data to the GPU and tracking the gradients reported from each mini batch. Each mini batch has a chance of randomly being selected to have a verification check performed. This verification once again uses Freivalds' Algorithm to verify the outcomes correctness. If any mini batch is not correct this is reported, and the model will not be verified as accurate indicating it was attacked. Right away its clear that this does not provide privacy protections for datasets so this method cannot be used on any sensitive data which limits the possible applications severely. There is no arguing that the randomized integrity checking can be an efficient and accurate approach; however, the combined overhead of this method plus anything else needed to provide other security aspects may be too great to overcome.

2.3 Multi-party Computation

Multi-Party Computation or Secure Multi-Party Computation (MPC) is a method for which two or more independent parties can share data for computations while keeping each party's data private from the other parties involved. This is a very attractive concept for ML on account of the fact that many separate parties may own portions of datasets that can be used to train a model but not a single dataset large enough to accomplish the task. Methods for MPC are purely cryptographic and the focus is mainly on data privacy.

Falcon introduces a 3-party protocol that has several advantages over other previous methods, Wagh et al. 2020. It is notably faster than other methods used as comparisons and claim to be the first to support batch-normalization. The main focus is to allow 3 parties to share data for ML

without any party getting the data of other parties. In terms of using the GPU for ML, no specific additional security is employed though they did execute performance tests to compare performance of CPU versus GPU using Falcon which gives some good performance insight, and the authors highlight the need to include GPU support for ML. Overall, security is still lacking in terms of the complexity encountered in a cloud environment.

CrypTen is another approach to MPC which focuses on bringing MPC into mainstream acceptance, Knott et al. 2021. The advantages here are ease of use and non-specific party numbers. It is also designed specifically to support use of the GPU in ML. The ability to cater to any number of cooperating parties is an advantage over other methods that are specific to 2 or 3 parties; however, the overhead in communication between the parties appears to get out of hand very quickly as the number of parties involved grows. There is a huge amount of privacy built in but there is not much mention of other types of attacks or vulnerabilities outside the cooperating parties. It's not addressing the issue at hand.

MPC may have great use in ML and seems like it may be a good option for certain use cases as it allows many parties to work together and keep data private from each other. It may be ideal in cases where data is coming directly from databases and is not readily available otherwise. There is nothing to indicate that this is a more secure choice over other methods for using the GPU to accelerate learning, it simply caters to multiple sources. Theft of model, poisoning attacks, and DoS attacks would still have the same virulence as in other software methods. In terms of considering data from multiple parties, there may be other options for this to be completed as well using the proposed method here which can provide greater levels of security if needed.

2.4 Limitations of Software-based Secure Machine Learning in the Cloud

Software solutions have the advantage of being quicker and more affordable to deploy when compared to a hardware solution for a great many problems. Dealing specifically with securely using an untrusted GPU for ML in the cloud, software solutions fall short in a lot of ways.

Each method seen so far is limited to primarily one security aspect, mostly privacy or integrity. These are issues that are well suited to be solved using just software via encryption,

hashing, and other methods. In the time required for ML, even these strong software-based concepts break down. Aside from homomorphic encryption, all the data must be in plaintext somewhere in the system to do work on which makes it vulnerable, much more so than a shorter process. But even the homomorphic encryption still needs an integrity check and may be limited in application. Each method is short of security features. This means that all the overhead added in each method is only a fraction of the overhead needed to provide a comprehensive software-based security implementation. This negates the reason to use an untrusted GPU which is to provide much needed performance increases over the long process of ML.

Software will always be limited to the hardware it runs on. In this sense, it can only be as secure as the hardware is. Looking at Intel SGX, clearly some serious exploits were found that allow data to be stolen that otherwise should be guaranteed secure by the hardware TEE. This does not mean that all TEEs will be insecure but at this time that technology may not be mature enough to be fully trusted. With software-based solutions, it is always about adding more code or overhead. Given any ML model to train, it will always be fastest training on a plaintext dataset on the fastest GPU. In order to secure this with software, more code must be added, a lot more. There will always be overhead that cannot be overcome except with a hardware-based solution.

CHAPTER III

HARDWARE SOLUTIONS

There are not a lot of hardware solutions currently available for this specific problem. TEEs such as ARM TrustZone and Intel Software Guard Extensions (SGX) are certainly hardware-based solutions except not specific to this situation and these technologies, for the time being, are not available on GPUs. Thus, solutions involving TEEs are primarily software based such that they leverage the security of a TEE on the CPU while using the GPU to accelerate the training process. A few solutions exist that have explored the options of making minor hardware changes to allow more secure use of the GPU for ML purposes. These solutions are still not comprehensive and may be too difficult to be practical.

The general consensus is that major hardware solutions to this problem are likely too expensive to be done. There is truth to this as the NVidia A100 80GB model is still listed at around $20,000. The price of such a device is justified by the diverse number of services that the GPU can provide for the cloud provider. With these things in mind any hardware solution would need to be very comprehensive while still allowing the same diversity in services without a negative impact. The following solutions require a form of modification of hardware in order to be implemented. In both cases, there seems to be no mention of possible ramifications of the implementation on the rest of the environment which makes them highly suspect to begin with.

3.1 Pure TEE Method

The TEE is a way to isolate data in a system by use of a CPU that is trusted. This is a hardware solution in because the system must have a CPU that supports TEEs, even though the TEE was not developed to specifically for use to solve this problem and adds no solution for using

the GPU in a trusted manner. For the purpose here we can say that not using the GPU at all is one solution in and of itself. We assume the processor is trustworthy because the manufacturer is trusted, and they provide the means of attestation such that the processor is of a known trusted source. Probably the most common source of TEE recently has been Intel SGX, the details of which are discussed in detail by Costan and Devadas 2016. Intel SGX allows for the creation of enclaves which in short are areas of memory that are encrypted and isolated by the CPU. This isn't physically isolated, just controlled separately by the CPU so that no data goes in or out of this area in plaintext and only the CPU can encrypt/decrypt the data. Intel SGX has been used widely in studies related to the untrusted GPU in ML problem.

Use of a CPU administered TEE is the slowest method for which to train ML models in the cloud. As such this becomes the baseline by which other methods are measured in terms of how much better performance is gained using methods involving a GPU rather than a CPU. For example, SLALOM, Costan and Devadas 2016, shows as much as 20 times increase in throughput compared to using pure TEE method.

Though it is generally accepted that the CPU will be slower than a GPU for ML this is not necessarily the case. SLIDE uses two 22 core Intel Xeon CPUs with a modified ML algorithm based on hash tables and they demonstrate that performance gains of at least 1.5 times over using a single Nvidia Tesla Volta GPU, Chen et al. 2020. The performance gains are impressive however there is no discussion of security. If only a CPU is needed, then a TEE security feature could possibly be leveraged for cloud use. This raises some questions. How much overhead will SLIDE incur if operating in a TEE? If the overhead doesn't far exceed the performance gains using this method, then it seems like this could be a practical solution. It has also been shown that SLIDE can be further enhanced by leveraging other new technologies available on modern CPUs that were not used initially. Daghaghi et al. 2021 have shown performance gains over 7 times that of the non-optimized SLIDE implementation using the same hardware. In terms of performance, this is hopefully encouraging enough to warrant an investigation into the practicality and usability of the method. Assuming this could be a legit TEE only solution, then the security of the TEE must be

considered.

At this time, the most popular TEE encountered is provided by Intel SGX. The answer to whether Intel SGX is secure enough for ML purposes appears to be a resounding no. Many vulnerabilities have been identified that make SGX an inadequate source of security. Cache-timing attacks have been able to successfully retrieve AES keys utilizing the built-in hyperthreading technology on Intel Processors, Götzfried et al. 2017. One can assume that if the AES key is retrievable then so is everything inside the enclave. Another cache attack was able to retrieve DNA sequences capable of personally identifying people from an enclave running preprocessing for genome sequence analysis, Brasser et al. 2017. This is very concerning considering the nature of ML and datasets. In a scenario where a pure TEE method is used for security, a cache attack such as one of these could leak an entire dataset and more.

One other attack which is particularly concerning is the SGX Bomb attack, Y. Jang et al. 2017. This type of attack effectively creates a local denial of service (DoS) attack within the system. This type of attack creates a situation where the system cannot proceed and is required to reboot. This in turn means that everything in RAM is lost. In regard to ML which uses large amounts of RAM and requires a long time to run, this type of attack could cause trouble from delays to completely rendering the environment unusable for ML purpose. In a competitive marketplace this type of attack could become a widespread problem. Other forms of attacks have been studied as well but this should be sufficient to demonstrate that SGX is not nearly as secure as it was meant to be. This may be why Intel has deprecated the Intel SGX technology from its 11th and 12th generation processors.

Intel SGX is not the only source of TEEs and going forward others are sure to be deployed. If Intel SGX is any indication of what to expect from other sourced TEEs, then the CPU must just be considered another untrusted piece of hardware which doesn't solve the problem. It leaves the CPU in the same situation as the GPU. Referring back to the SLIDE method, without the possibility of using the TEE to secure SLIDE, there is probably no practical advantage in a cloud environment since it may be unlikely to dedicate multiple CPUs for the process and there is no way to secure it.

This also means that not only are pure TEE methods insecure, but any method that relies on SGX for any partial TEE solution has also lost its security.

3.2 Isolation by Modified Interconnect

One proposed solution is the HIX method, I. Jang et al. 2019. This solution requires hardware modification and corresponding software to use the GPU securely, without any modifications to the GPU device. It requires the use of two SGX enclaves and a modified system to work. The general idea is that the drivers for the GPU are removed from the operating system and placed inside an enclave to operate the GPU from inside the enclave. Then the second enclave serves as a host to the application to be run using the GPU. Additional SGX instructions are needed and the PCIe root complex must be modified in this case for this solution to work. There are a lot of issues with this solution. The solution was tested using virtual machines in which the system could be modified but in practice modifying the system maybe unrealistic. HIX has a fair number of limitations and would have to be refactored to work without SGX at this point it seems. It still does not cover enough security aspects to be a full solution and incurs a fair amount of overhead. They reported a 26% loss in performance over an unsecured GPU versus HIX implementation which is a considerable loss for a solution that may still leave a lot of vulnerabilities.

3.3 Emulated GPU TEE

Graviton, by Volos, Vaswani, and Bruno 2018, is another solution that is somewhat similar to HIX. In this solution, SGX is once again utilized but in this case only to secure an emulated GPU TEE. A few modifications are required for this to be achieved. The PCIe control engine and the GPU command processor need modifications for this to work. The takeaway is that they were able to implement an emulated GPU TEE on a GPU that has no hardware support for it using modified drivers and encryption along with the hardware modifications. Many security assumptions are made with this method, and it does overlook other major security issues. It incurs 17 to 33% overhead while combined with the necessary modifications to make it work practice make it a less than ideal solution and certainly not comprehensive.

3.4 NVIDIA H100 PCIe Gen 5 GPU

The latest GPU solution coming from NVIDIA is the H100 Tensor Core GPU. It's no surprise that there are considerable upgrades in performance and scalability which will aid in increased ML use cases. The real important updates that pertain to the situation here are the new security features. Of course, this product is not on the market just yet so there is no hard data to work with but the details from the white paper look promising.

Probably the most major upgrade is the inclusion of support to make a dedicated GPU TEE. This looks to be the first official support for TEE on a GPU which is a huge advancement as prior to this TEEs have been exclusively for CPUs. The wording is vague, but it looks as though this is accomplished by giving the GPU the ability somehow to create its own TEE which may be similar to how ARM TrustZone works. The exact details about how this will work are still fuzzy at this point, but this make sense as NVIDIA was recently interested in acquiring ARM from SoftBank Group which did not happen. NVIDIA is highly invested in ARM technologies and will feature a CPU + GPU super chip consisting of two ARM CPUs and one H100 GPU combined though this may not be available on all H100 products.

Other security features include Measured Boot, Root of Trust, Device Attestation, and AES-GCM-256 Encryption of data between to and from the GPU. Measured Boot provides a verification that the system is in a secure and ready state by reporting elements of the boot process. The on-die Root of Trust validates the firmware such that it is authentic and not modified. Device Attestation further ensures that an authentic NVIDIA GPU is being used with authentic firmware. The AES-GCM-256 encryption provides fast transfers of encrypted data between the GPU and CPU ensuring that data is kept private in transit.

All around there is an impressive amount of security built into the H100 and in truth this may be the total solution already. Almost all of the requirements previously discussed are covered leaving only interruptibility truly not accounted for in some form. The main question will be how secure is the provided TEE for the GPU? Intel SGX has already been shown to have many security flaws and data has been extracted successfully. ARM TrustZone has also had its share of successful

attacks which, as per given speculation, might be the basis for the TEE on the H100. Once deployed on real world environments, time will tell if this GPU can provide the necessary security. Regardless it is certainly a major step in the right direction.

3.4.1 Other H100 Devices

The H100 GPU is or will be used in a variety of other products and forms. These other devices are out of the scope of this paper. This includes all the devices are not PCIe standard cards that fit mainstream servers in standard racks. The H100 SXM5 GPU, DGX H100 systems, HGX H100, and the H100 CNX may all need a different solution if this specific use case for them were to arise. There are plenty of other services that benefit from the acceleration of using GPUs, so there are most certainly situations that simply do not require the degree of security, if any, needed for this specific use case.

3.5 Limitations on Hardware-based Secure ML in the Cloud

As mentioned before, hardware solutions are likely the most expensive solutions and can be very difficult to implement. Hardware solutions need to be engineered, prototyped, tested, built, and deployed in a timely fashion such that they are not irrelevant upon hitting the market. All software is dependent on the hardware to enable its usage so there is likely very few problems that cannot be solved by using better, newer, or more specific hardware.

The majority of limitations on hardware fall into the category of feasibility issues. Assuming any needed piece of hardware can be made on a large enough budget then the real limitation is financial. Hardware solutions need to be financially viable in order to be considered for implementation. Looking back at some of the examples, different hardware components would need to be replaced in order to implement the given solutions. To get a better TEE out of Intel SGX, the processor would need to be replaced with one that has different functionality. Well, to be able to replace a processor, the socket will need to be updated. This means that all the main boards need to be replaced with new boards with updated sockets. It is inevitable that all data centers would eventually need do this, but not until the end-of-life cycle of the hardware components. Modifying something like the

PCIe bus may take even longer. That is a standardized bus that is held to tight specifications and the processor must be able to support that version of PCIe. To implement a hardware solution to a single specific problem like using an untrusted GPU for ML, that solution must require the minimal number of modifications possible and limit those changes to devices that can be easily added to a system otherwise the changes to the server hardware environment would be far too costly. Sure, companies can wait till the inevitable hardware upgrades are required but that isn't how business works most of the time.

One severe limitation on hardware is that, in most cases, in order to do work on any data, that data will need to be in plaintext at some point. Whether it be just in a cache or in registers. This is unavoidable and probably the lowest level of security issues. Intel SGX does not work well with Intel Hyper-Threading Technology which creates an easy path to get data from an enclave because in cache that data is in plaintext as it must be. This shows that another level of isolation is needed to truly keep that data safe. Despite any of these limitations, hardware-based solutions are going to be the best option. It just needs to be done correctly.

CHAPTER IV

FIGHTE: FULL ISOLATION OF GPU HARDWARE FOR TRUSTED EXECUTION

4.1 Introduction

With all the current options for securely using a GPU in a cloud environment for ML, none of them so far offer a truly comprehensive plan for security while also being a feasible implementation. Many of these solutions lean heavily towards privacy while not addressing other issues present in the environment. These solutions may have their application in some situations but with the growth of data sets and the rising need for greater accuracy the need for truly secure and trustworthy cloud ML options is culminating. Evidence to this can be seen in the newly announced GPU from Nvidia, The Grace-Hopper Superchip NVIDIA 2022, which brings a new level of security to ML along with an impressive number of enhancements all around. Though this has not reached the market yet, it looks to be a major advancement towards a solution to this particular problem. It may have its shortcomings as well, but it certainly is a leap forward in terms of security and already has major features that FIGHTE would require to make possible.

In short, the solution here is to provide physical isolation by creating or modifying an existing GPU card such that a subsystem on a card can be created and run independent of the main system. The subsystem will have all the necessary components needed to perform the tasks given and will be able to run without any other form of security once isolated thus ensuring the highest performance achievable. There is no arguing the security of isolation which is why TEEs are a popular idea. This is the same ideology in that data is encrypted in and encrypted out but not software implemented so there is a level of isolation that software cannot obtain. Obviously, no one cannot go and physically remove devices in a data center that have running processes on them, so

the goal is to add physical disconnects and security checks to the GPU cards in order to verify the hardware is isolated.

4.2 Requirements

A number of security issues have already been highlighted and each will need to be covered to provide comprehensive security. This includes side channel or cache attacks as well as DoS type attacks. Other possible scenarios that have not been discussed may benefit from this method also. Hardware trojans have been a hot topic in security recently as this may be something difficult to detect or protect against if deployed. Here, a list of general requirements to create a highly trustworthy environment for which to run long ML processes is given and discussed. A summary of the requirements can be seen in Table 4.1.

Table 4.1: List of Security Requirements

Requirement:	Reasoning:
Must protect data privacy at all times.	Prevent theft of data or private information contained in data.
Data and model must not be modified.	Maintain integrity of the model and data.
Hardware component must be verifiable.	Must verify the legitimacy of the component for trust.
Must not be interruptible.	Denial of service attacks could lead to lost time and possible other damages.
Modifications contained to one piece of hardware.	Replacing too many components is not feasible.

4.2.1 Basic Requirements

Some of the security requirements of course are the same for security in general. The goal is to follow the CIAA ideology; being Confidentiality, Integrity, Availability, and Auditing, without being concerned about availability as that is not in within the scope. Maintaining data privacy, as seen in earlier examples, has many possible solutions. Generally, modern encryption standards should be suitable to prevent any breach of privacy. Maintaining and testing for data and model integrity is another rather trivial issue. Combining cryptographic schemes with hash value testing to

ensure that nothing has been modified at various points will ensure no modifications are made to the data set prior to or during the training and testing process. The model of course will be expected to weather changes, but integrity checks may still be an option particularly for transit purposes.

4.2.2 Attestation

Hardware verifiability is needed to ensure that the hardware components expected are indeed the ones being used. Attestation is the method to which the hardware can be verified as legitimate. This is achieved by using public-key cryptography. Each GPU device will have a serial number or a similar device identification (ID) to identify it uniquely. With each uniquely created piece of hardware, a private key is issued that is hard programmed onto the device. The public keys for all devices are then made available to clients that will be using the devices. Using a device ID such as a serial number, the client should be able to obtain the public key for the device which will verify that it is indeed the expected piece of hardware. Combined with integrity and verification of the firmware or software of the device, this will provide significant evidence for trustworthiness of the environment to perform the ML task in.

4.2.3 Modification and Design Requirements

These requirements are designed to minimize the costs and impacts of implementing a solution. The data center hardware environment is complex and expensive. It is very precisely engineered. Every component has its size and shape, and these tend to stay the same through various hardware iterations. It bodes well for any solution such as this which addresses one specific problem directly to follow suite.

Restricting changes to one piece of hardware will prevent the need for overly drastic changes to be made which would likely shelve any idea of a hardware-based solution. In this case, the entire solution should be on card with the GPU. This allows for upgradeability and scalability with current hardware in any environment without changing or modify servers entirely.

Keeping with current design formats will assist in making this solution acceptable. If the modifications are only made to the GPU card, then the GPU card needs to stay basically the same

dimensions as current cards. Typically, in a data center there are clusters of GPUs mounted together so each cluster of these modified GPU cards needs to be the same as any cluster it would replace. This may seem obvious but when considering adding considerable new functionality to a component, it may be difficult to fit all the necessary new components on the one card in its current size and shape.

4.2.4 Interruptibility Requirements

Though this may be something difficult to guarantee, it is still an issue that needs to be addressed. As mentioned previously, the ML process can take considerable amounts of time. Days, weeks, and even possibly months in some extreme cases. Considering that there are already DoS attacks on a hardware level as seen in Y. Jang et al. 2017 where the system is forced to reboot to recover from an attack, it would be wise to expect that such an attack could be carried out against a co-processor or GPU. If this were to be the case, then considerable amount of time could be lost in order to recover from such a reboot, or worse, data corruption could occur. Whereas it is never guaranteed that an interruption cannot occur, it can certainly be minimized substantially by eliminating as much as possible the threat of this occurring maliciously.

4.3 Isolation Concept

As mentioned previously, the goal is to provide physical isolation for a GPU to perform ML tasks in the cloud. There are different possibilities for which this could be accomplished. Here the focus is to provide a single PCIe card-based GPU that has the capabilities needed to isolate itself physically while still remaining physically attached to the rest of the system. In this manner we reduce all the modifications in hardware to a single card that can be added to a system or replace a current GPU device. No other system modifications should be needed.

4.3.1 GPU Modifications

Quite a few devices will need to be added to any current GPU in order to make this possible. A summary of these devices along with the purpose for each is available in Table 4.2. Also, power will need to be a consideration as we are trying to physically isolate the GPU but still keep it

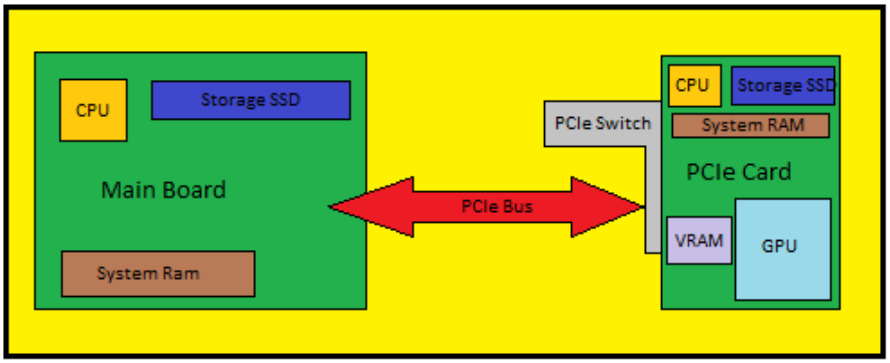

Figure 4.1: Concept View of Modified GPU Card to Act as a Standalone Subsystem

operating. Most GPU cards get power from both the PCIe card slot and a power cable from the power supply. PCIe card slots can provide approximately 75w of power where most cards will require around 350w. It may be prudent to simply remove any reliance on power from the PCIe slot altogether for this usage though that might be a challenge. Fortunately, PCIe is very general purpose. A lot of different devices can be run in PCIe slots. In a typical system, GPU cards, sound cards, networking cards, storage SSD cards, and many others operate in the same PCIe slots. With that, there is no reason that any changes should need to be made outside of the GPU card itself. A quick view of the changes needed to the PCIe card can be seen in Figure 4.1.

Table 4.2: List of Devices for GPU card

Device:	Purpose:
CPU	Control the GPU and other general purpose usage
RAM	Need main memory for on-board CPU
Storage Device	Must have rather large capacity storage device to store dataset
Physical disconnect device	Create the isolation of the hardware subsystem
Physical hardware environmental sensors	Detect tampering of the hardware

There are certainly cases where multiples of GPUs are used to accelerate ML processes. In this solution, the goal is to solve the problem simply for one GPU. Scalability is very important but if the solution works for one instance it can be reused and redesigned to adapt to the scale needed.

So, the focus is to build a PCIe GPU card with everything needed to be a standalone subsystem that can be set to task and not be susceptible to attacks of near any kind. A typical GPU card does not have the devices built on to handle this. Generally, a GPU is controlled by the system CPU. It has its own VRAM, which is used for GPU processing, but it gets its data from the main memory via direct memory access (DMA). In order to change this, those main devices located on the main board will need to be added to the GPU card along with the other supporting devices.

Addition of CPU and RAM Adding the CPU and RAM for the CPU are the most critical additions for standalone functionality. This will allow the PCIe card to be a system by itself. A CPU cannot function without RAM, these two go hand in hand. Getting this onto the card may be an engineering challenge but considering smart phones now have up to 12gb of LPDDR5 RAM its reasonable to expect that a decent amount of ram can be added to a device so much larger than a smartphone. The CPU itself should be relatively small. The ideal choice would be an ARM, which we already know pairs well with LPDDR5 RAM and is a very low power device, compared to other choices such as Intel or AMD. The focus is to have the GPU do the workload and the CPU take on the supporting role.

Storage Device Another necessary component will need to be a storage device to hold all the data needed. Ideally this should be an SSD. Fortunately, recent SSDs are relatively small and are available in multi-terabyte capacities. Non-Volatile Memory Express (NVMe) is the current standard for the fastest SSDs and is basically a type of PCIe interface. There are various form factors in use for NVMe SSDs but in this application an M.2 slot SSD should be able to provide several terabytes of storage in a small space. In the even that much more memory is needed an E1.L slot SSD could be used which could provide over 10 terabytes of storage and still be small enough to fit on a GPU card. Overall, it should be very possible to add an SSD to a GPU card without increasing the size just using the available slots, not considering the possibility of adding the SSD directly to the card.

PCIe Disconnect In order to achieve isolation, some form of disconnection must be made. A possible method would be to build a switch onto the GPU card that is controlled by the onboard CPU. This switch would allow the CPU to set the isolation once all data is received and maintain the isolation throughout the training process. This switch may disconnect all or some of the PCIe lanes. It would seem than an efficient approach would be to only disconnect the lanes that may allow input and output of data such that only data isolation is achieved. However, total isolation is achieved then the ability to weather through situations where the main board reboots while the GPU is running without interruption. Ideally then, the best approach is completely switch of all PCIe lanes. This also would mean that any reliance of power from the PCIe should be removed as well. Most GPU cards require additional power source outside of the PCIe anyways so it may be reasonably easy to simply get all power from the power plugs.

Environmental Sensors These sensors should simply be switches used to detect tampering. Some devices may already use these types of sensors. The goal is to detect tampering of the device while the hardware is isolated. A typical GPU card in the cloud environment has only the PCIe male port and a link to connect to other GPU cards such as NVIDIA's NVLink. There are no HDMI or DisplayPort connections built into these GPU cards. This means there are minimal ways for data to get in or out of the card and these paths need to be monitored. A built-in series of switches should be able to detect if a device is added or removed from the link port, or if the card is removed from the PCIe port. These switches could alert the card of a breach so that it can react and protect data. Other switches may be located on other devices such as the

4.4 Data Flow Through Hardware

Since this solution requires that all the data be located in the isolated card, a fair amount of data transferring must occur. The flow of data through the hardware components is a reasonably simple concept itself as can be seen in Figure 4.2. All the data will be located in the main system storage or in transit to the main system and will be encrypted. It will stay encrypted until it is completely transferred to the subsystem and completely isolated. Once there and verified, all tasks

will be performed in isolation. At the end of the training and testing of the model, the data set can be deleted, assuming it was only a copy, and the model can be encrypted for return to the client. Then the card can reestablish a connection with the main system and deliver the model.

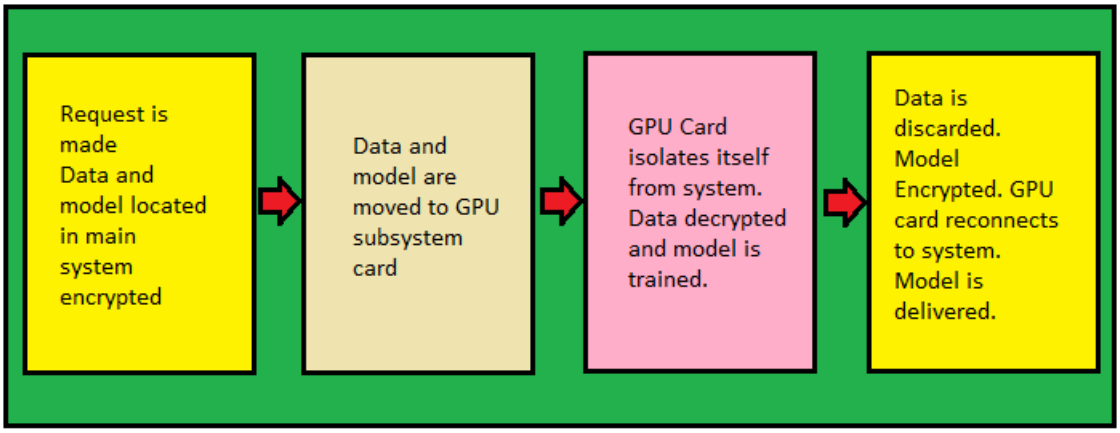

Figure 4.2: Flow of Data through the Hardware

One thing that stands out from this is that there may be multi-terabyte data sets being transferred across the system and subsystem card. This raises the concerns over how much time might be needed to transfer this data and where the bottlenecks might occur in data transfer. In the past, the idea of transferring a TB of data might have sound absurd but with today's technology transfer speeds have increased dramatically. Looking at Table 4.3, a comparison of various bus speeds, the slowest given is the 1 gigabit ethernet LAN. This is the slowest point and actually a lot faster than most business and household internet services available from providers. Certain regions with access to fiber optic infrastructure may see speeds this high or higher but not the vast majority of users. This however is just getting the dataset to the cloud provider and the dataset may come from different sources so that does not affect the times of the service itself. Looking at the other speeds listed, any modern data center should be using PCIe based SSDs. M.2, U.2, and other form factors can be implemented on the PCIe bus and use PCIe speeds, typically at PCIe x4. There will likely be a bit more time to transfer data in the real world than the theoretical max speeds of these busses, but the internal bus speeds show that that the time to transfer large data sets can be measured in minutes emphasizing that this isn't the time-consuming task it would have been in years past.

Table 4.3: Comparison of Various Common Speeds

Transfer Bus:	Theoretical Speed:	Transfer Time of 1 TB of Data
1Gb ethernet LAN	125 MB/s	2 hours, 13 min, 20 sec
SATA SSD read speed	500 MB/s	33 min, 20 sec
SATA SSD write speed	500 MB/s	33 min, 20 sec
M.2 SSD read speed	7 GB/s	2 min, 22 sec
M.2 SSD write speed	5 GB/s	3 min, 20 sec
USB 3.2 2x2	2.5 GB/s	6 min, 40 sec
PCIe 4 x4	4 GB/s	4 min, 10 sec
PCIe 4 x16	32 GB/s	31 sec

4.5 Software Implementation

In order to utilize the modified hardware, management software will have to be written to accommodate the isolation aspect. There will need to be 2 parts to the software that will work together. The first part will remain on the main system as the primary component. This will be the program that communicates with the client, gathers data, and prepares and verifies the subsystem to do the training. The secondary part will be what operates on the subsystem card. This portion will lockdown the subsystem, maintain the isolation, and perform the ML training. Once the training is completed it will re-connect with the primary program.

4.5.1 Primary Software

The primary program will take on most of the responsibilities of the overall process. This part of the program will communicate with the clients, prepare data for processing, and verify the initial setup of the hardware as well as the legitimacy of the hardware involved. These responsibilities should include verifying that data used for training is from a trusted source, is encrypted, and has an accompanying encrypted key and hash value, as well as collecting all the parameters needed for the training to run to completion.

With all the data received and parameters set, the program should run the attestation to verify the hardware and firmware of the subsystem card. Once verified, the program will load the data to the card and configure the secondary program on the card. A final verification should be

performed to verify the subsystem is configured correctly, then if verified, the primary program will hand off control of the subsystem to the secondary program. The primary program will wait for the secondary program to re-establish communications. After the secondary program is finished the primary program will regain control and establish or wait to establish communications with the client to deliver the trained model. A general flow of the primary program is shown in Figure 4.3.

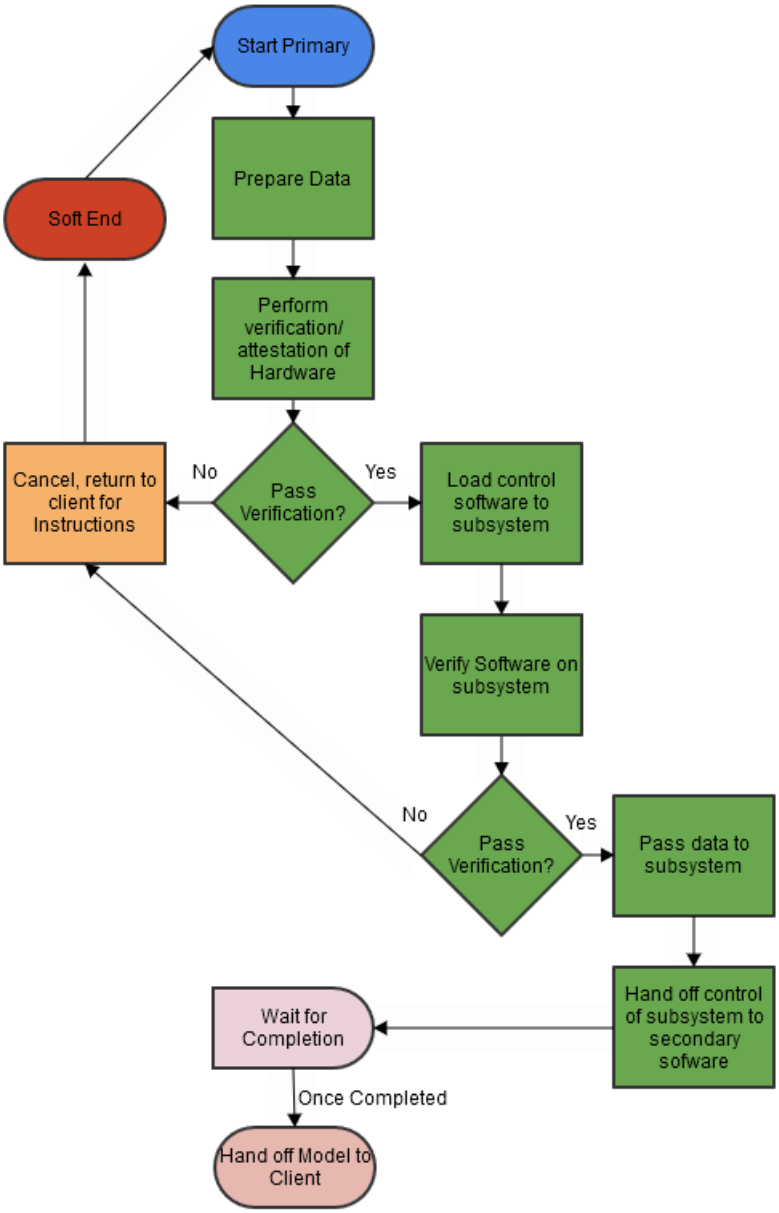

Figure 4.3: Workflow of Primary Program.

4.5.2 Secondary Software

The secondary program will be only on the subsystem card. This will be responsible for maintaining the isolation of the card as well as coordinating the training and testing of the ML model in isolation. The general flow of the secondary program can be seen in Figure 4.4

Once the primary program reaches waiting stage, the secondary program running only on the GPU card will lock down the card and initiate a monitoring process to make sure the card is not tampered with during the training and testing of the ML model. At any point, if tampering is detected, the data can be wiped or encrypted, and the entire training process aborted, and control returned to the primary program. After the card is successfully isolated, another verification should be run internally to verify no changes were made prior to locking it down. After the isolated environment is confirmed internally, the process of training the ML model can commence at native speeds in a physically isolated environment. Once the training is finished, the model can be encrypted, and the dataset discarded as it should only be a copy of the dataset. Memory can be wiped such that the data anything sensitive that had been decrypted is surely gone. Once all training is completed based on given parameters and transit level security restored, the secondary program will return the card to an unlocked state and control can be returned to the primary program. Once control is returned to the primary program, the subsystem card can be wiped clean of all data and set to a ready state.

4.6 Implementations and Testing

The costs and time frame for manufacturing an actual device to test for this solution are obviously prohibitive. No such devices readily exist or are easily modified to use a placeholder for testing therefore no tests have been performed. This is likely a solution that would best be given to a major GPU manufacturer that has the resources to prototype and test such devices. That does not mean it cannot be done by other entities. Certainly, GPU manufacturers sell the GPUs for use on PCIe boards that are manufactured by various other companies that do not directly manufacture GPUs, so it would be quite possible for a company to take this solution and apply it to an existing GPU on a custom made PCIe card.

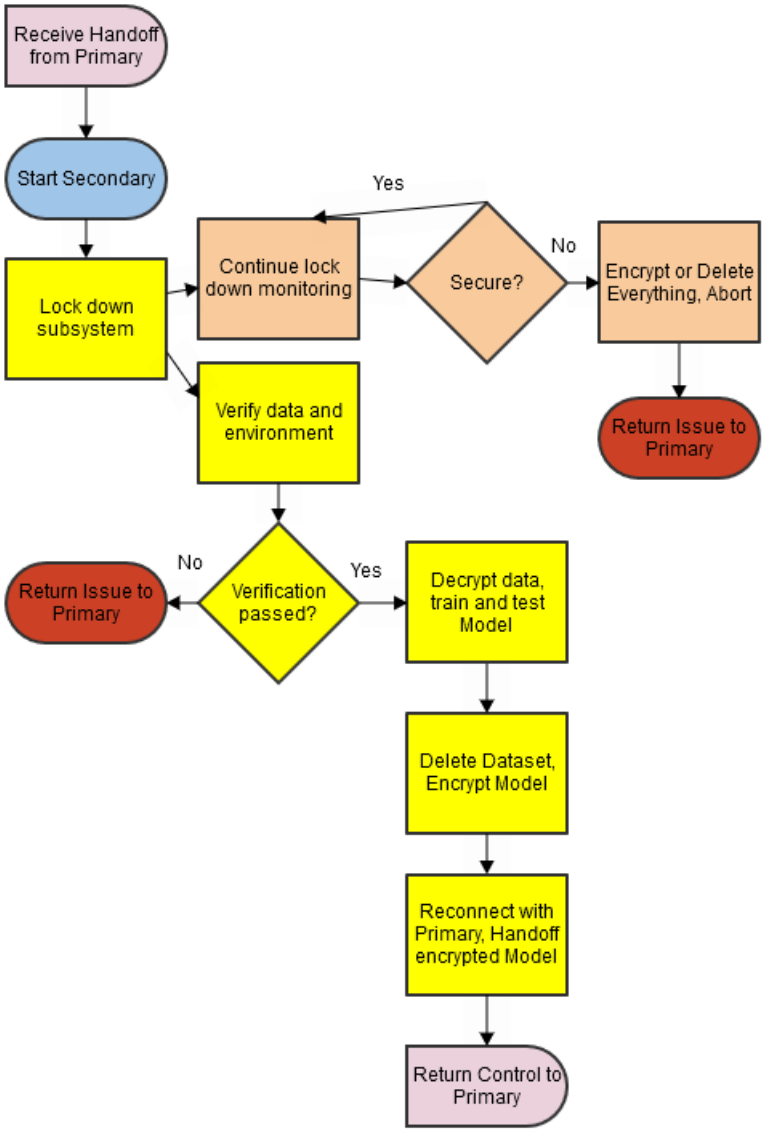

Figure 4.4: Workflow of Secondary Program

There would be quite a few engineering challenges to get this done. Placement of all the components on the board with adequate connections and cooling capabilities would be one challenge certainly, though probably not the most difficult. Maybe the biggest challenge would be able to create the physical disconnection. Physically switching off the lanes is as easy as installing a switch, but the device needs to be able to switch off the PCIe lanes without disrupting the system. The question becomes how can this be done in a manner that is supported by PCIe or does not cause any issues with the interconnect? On a typical computer, PCIe resources are allocated when the

computer is booted and really that can't be changed without modifying the operating system and/or the BIOS/UEFI on the motherboard. However, PCIe on server boards should typically support hot swap or hot plug capabilities and these would be the first options to explore. In essence, set the card to be swapped, switch the card into isolation mode, run the ML programming, and then switch back on the PCIe lanes and add the card back to the system. If that were to prove to not be a viable solution, then some sort of PCIe spoofing should be doable that keeps the system informed that the card is still there, but the processors are physically isolated on the card behind the switch.

Another challenge to get this implemented will be all the software needed to support the device. A new set of device drivers will surely need to be written to support the card as well as a new application programming interface (API). Depending on the devices used some of this may be created from existing drivers and API that need only be modified and updated for use in this application. These may not be exceedingly challenging but will certainly add to the costs of implementation.

4.7 Security Implications

Testing an actual GPU card for results would be unnecessary to accept the validity of the security concept provided by this solution. Physically isolated devices provide a degree of security that cannot be matched in software methods. Other issues such as encrypting data and passing encryption keys securely are trivial in that they are widely performed already indicating that the security presented is mature and easily trustworthy.

A few assumptions are made about security. It is assumed that cryptographic keys will be enough to keep data secure and while used alongside hash values the integrity can be verified as well to ensure no changes are made in transit or otherwise while not isolated. Another assumption is that all parties that provide data are trustworthy. In the event that there is a malicious party declaring itself as being trustworthy, FIGHTE is not going to be able to defend against that. These are details that should be worked out prior to attempting this task by all parties involved. Of course, with proper use of certificate authorities and other identity verifications, there should be plenty of protect from outright adversaries posing as a known trusted party. Since a great part of the security relies

on the hardware manufacturer, it is also assumed that they will manufacture and provide attestation services in good faith such that no designs are made that would allow the manufacturer to steal data. There is also the possibility of a hardware trojan. Since this level of security would be very difficult to beat, it could make an isolation device a prime target for a hardware trojan. This calls for a further assumption that any part of this device is manufactured by a trustworthy source. Ultimately, construction of such devices would have to be under great scrutiny.

In regard to specific threats against using an untrusted GPU for ML in the cloud, we can clearly see the level of security this will provide. Facing the issue of data leakage or theft of data, this is heavily prevented. All data remains encrypted until it is isolated physically. Even theft of the encryption key would be extremely unlikely. Given that a GPU card matching the given requirements is created, it would certainly use public-key encryption to encrypt a symmetric encryption key used to decrypt the data for use in training. The card would have a specific private key built in unique to that card and the public would have the public key to encrypt other keys with. Theft of model is prevented as the model is kept encrypted as well. There could be no opportunity for side-channel attacks or cache attacks. The co-processor and GPU are isolated so no adversary can view the cache or share threads on a core. Certainly, all the data in the RAM and VRAM could be in plaintext but no one could see it. No one would not be able to physically steal the card and get that data from it because once power is removed that data in RAM is lost and data in storage is going to be still encrypted. If implemented with no dependency on the PCIe bus for power, then there is a huge protection from interruptions based on DoS attacks. While maintaining independent power and isolation from the main system, its quite possible for the main system to be rebooted, multiple times even, and the main program recover back to the waiting state. Clearly, physical isolation of a process like this is a very robust solution.

4.8 Performance Implications

Performance of this method should be unmatched compared to other methods. The only additional overhead incurred for this method is additional transfer of data onto the subsystem card and the verification checks. The data transfer times have been shown to be trivial amounts of time

and it would be difficult to even measure how little time the verification checks might add to the process. If we factor time for decryption this should be neutral time increase as other methods require data to be decrypted as well and decryption should not be a time factor as is since the data arrives encrypted and needs only be decrypted as the training process commences. In this scenario, the performance of training the ML model is only restricted by the performance of the hardware itself without any additional overhead required for security purposes.

4.9 Other Challenges and Limitations

A number of limitations may exist for FIGHTE. In particular, this method may not be able to be applied to all types of ML in the cloud. The focus in FIGHTE is to provide security to deep learning processes where the data is fully available in terms of how it is delivered. This means that all the data needed can be packaged and delivered to the GPU card where it is then isolated, and no other data is needed to proceed with training. This in practice might need to be changed to accommodate bringing in new data at intervals. In the case that a model needs a direct stream of data another solution will need to be found. It may be that there are models that are constantly changing to adapt to new data as it is created or becomes available which may place high availability constraints on the model preventing it from being capable of being isolated. This would create significantly more difficult security situation for ML in the cloud altogether.

One challenge may be to include catering to multi-tenant usage. In the cloud, there are lots of tenants on the same hardware including the GPU. In this case we are isolating an entire GPU for one purpose. It is assumed that the entire GPU would be needed but this may not be the case. Since FIGHTE is isolating the entire GPU, it will not be available to any other tenant for any other purpose until this process is complete. Of course, there would likely be other GPU devices available from the provider which can serve other tenants it may be a consideration that too much demand on GPUs that are isolating may leave other services under served. It would not be a trivial task to try and serve multiple tenants on one GPU that is physically isolated even if this were to be a possibility to load up multiple of datasets and models for various clients to train concurrently using a single GPU device.

4.10 Possible Drawbacks of this Method

Certainly, one of the most formidable drawbacks is the expensive of manufacturing the card. A large degree of design and engineering will need to be done to produce the card and then testing will need to be done to ensure proper functionality. Whereas the cost alone would not rule this out as these cards are exceedingly expensive already, there must be a demand for this level of security. The price paid for the GPU cards may be justified by the number of services that can be provided or benefit from the acceleration provided by the GPU. Cloud based Gaming, Data Analytics, and Scientific Computing are just a few of the other services this type of device caters to. In the sense that this modified card may be overly application specific, it might be the case that a lack of diverse uses cases renders the cost insurmountable.

4.11 Comparison with NVIDIA H100

The NVIDIA H100 is set to be released in 2023. There are still a lot of details missing such as exactly how the TEE will be implemented for the GPU. The descriptions and specifications given so far are very general. It appears that the TEE is handled by the GPU itself, though it's likely a CPU on the main board handles another TEE of some sorts for other aspects. It is stated that the H100 will be usable with current x86 architectures so assumingly it can be paired with other CPUs using other technologies such as AMD's Secure Encrypted Virtualization (SEV). What is known for certain is that there will be a Grace-Hopper Superchip that uses NVLink-C2C (chip-to-chip) interconnect to connect 2 Grace ARM based CPUs with a Hopper GPU. Whether or not NVIDIA places a CPU onboard the PCIe Card for the H100, it is known that this configuration can be done, possibly with a single less powerful CPU. The NVLink-C2C technology solves at least one technical issue of making it possible to have the CPU collocated with the GPU on a card that could be physically isolated. NVIDIA advertises NVLink-C2C as an interconnect for custom and semi-custom designs.

In terms of security, the goal of the Nvidia H100 is the same as FIGHTE in that the end result is the isolation of the data in use. Also, both methods are concerned with isolating the entire GPU for the duration of the training process. NVIDIA uses the TEE option and extends this to the

GPU offering the ability to isolate an entire GPU for one TEE or allow multiple tenants to use the same GPU in TEEs. Certainly, scalability is prized and for such times that clients do not need the full strength of this exceptionally powerful GPU this is a big plus for the TEE method. Of course, in cases that do or do not need the full GPU but need the full isolation to reduce risks, NVIDIA has the option to isolate the GPU entirely as well. FIGHTE obviously isn't factoring in scalability. If scalability were a factor for FIGHTE, the scale would start at isolating a single GPU and then scale up only in terms of number of GPUs. There would be no way to serve multiple clients on the same GPU because it requires the entire GPU to be physically isolated. The level of security in FIGHTE is intrinsic, physical isolation need not be proven excessively to be accepted. TEE isolation is based primarily on cryptography which is very secure but may be compromised if the encryption key is lost. This has happened on Intel SGX already. The TEE isolation will need substantial real-world testing to prove it can create a legitimate degree of isolation that cannot be defeated.

The Device Attestation feature NVIDIA has included on the GPU as well as the On-Die Root of Trust provide key features that would be needed for an implementation using FIGHTE as well. As discussed previously, it is critical to FIGHTE to be capable of verifying that the hardware and software on the physical device to be isolated are legitimate devices running known safe software and drivers. NVIDIA's Device Attestation and On-Die Root of Trust do just that. The H100 will also feature Measured Boot, which ensures that the GPU device is booted using only approved code and firmware to boot into a secure state. NVIDIA has a list of other security related enhancements included on the H100 that have not been discussed in detail as of yet.

Overall, it seems that NVIDIA is making a massive effort to solve the problem of using GPUs securely in the cloud for any purpose, especially ML, while at the same time bringing huge performance enhancements. It may be too soon to tell if the security enhancements will be enough. Specifically, the question is how robust these new TEEs on the GPU will be when up against attacks. If all goes well for the H100, it may very well be the end of this problem. If not then another more secure option may be needed, which FIGHTE certainly could provide.

CHAPTER V

CONCLUSION AND FUTURE RESEARCH DIRECTIONS

5.1 Conclusion

The proposed method allows for full speed usage of a GPU in a cloud environment where the attack surface is broad due to the complexity of the multi-tenant environment. If implemented correctly, the isolation will guarantee that no theft of data, or tampering or poisoning of the ML model in anyway can occur during the lengthy training process. It also provides a new layer of protection from interruptions that occur due to malicious behaviors such as denial of service attacks. The concept of physically isolated hardware components is undoubtably an insurmountable layer of security and should represent the apex of security in any computer system. With upcoming advancements in security related to GPU devices, it is clear that major hardware manufacturers are aware of this problem and are looking to for solutions that will enable them to tap into a growing ML and AI market that requires exceptional security and trustworthiness to handle sensitive and valuable data.

5.2 Future Research

One certain aspect for future work is to research devices that can be used for or modified easily to be used as a proof of concept for this approach. Time and financial constraints have rendered this not possible for this paper but with some creativity a testable solution is sure to be found.

With the newly announced upgrades in hardware coming from NVIDIA, it will certainly be a primary focus to follow up with news and information relating to all the H100 products up to and beyond the release of the products. Most notably it will be of great interest to follow

up with evaluations of the level of security provided by the GPU TEEs and possibly test that security firsthand. If any major security breaches occur, then the concept of the trusted execution environment may have to be revised or replaced with something more fundamental with concrete guarantees such as the approach given here. With that in mind, seeking new methods to apply this concept of hardware isolation to other related devices in the cloud infrastructure that might be performing the same tasks such as the variants of the H100 will be a focus as well in order to present this concept in as much of a realistic manner as possible.

REFERENCES

Asvadishirehjini, Aref, Murat Kantarcioglu, and Bradley Malin (2020). "GOAT: GPU Outsourcing of Deep Learning Training With Asynchronous Probabilistic Integrity Verification Inside Trusted Execution Environment". In: DOI: 10.48550/ARXIV.2010.08855. URL: https://arxiv.org/abs/2010.08855.

Brasser, Ferdinand et al. (Aug. 2017). "Software Grand Exposure: SGX Cache Attacks Are Practical". In: *11th USENIX Workshop on Offensive Technologies (WOOT 17)*. Vancouver, BC: USENIX Association. URL: https://www.usenix.org/conference/woot17/workshop-program/presentation/brasser.

Chen, Beidi et al. (2020). "SLIDE : In Defense of Smart Algorithms over Hardware Acceleration for Large-Scale Deep Learning Systems". In: *Proceedings of Machine Learning and Systems*. Ed. by I. Dhillon, D. Papailiopoulos, and V. Sze. Vol. 2, pp. 291–306. URL: https://proceedings.mlsys.org/paper/2020/file/65b9eea6e1cc6bb9f0cd2a47751a186f-Paper.pdf.

Costan, Victor and Srinivas Devadas (2016). *Intel SGX Explained*. Cryptology ePrint Archive, Report 2016/086. https://ia.cr/2016/086.

Daghaghi, Shabnam et al. (2021). "Accelerating SLIDE Deep Learning on Modern CPUs: Vectorization, Quantizations, Memory Optimizations, and More". In: *Proceedings of Machine Learning and Systems*. Ed. by A. Smola, A. Dimakis, and I. Stoica. Vol. 3, pp. 156–166. URL: https://proceedings.mlsys.org/paper/2021/file/3636638817772e42b59d74cff571fbb3-Paper.pdf.

Götzfried, Johannes et al. (2017). "Cache Attacks on Intel SGX". In: *Proceedings of the 10th European Workshop on Systems Security*. EuroSec'17. Belgrade, Serbia: Association for Computing Machinery. ISBN: 9781450349352. DOI: 10.1145/3065913.3065915. URL: https://doi.org/10.1145/3065913.3065915.

Jang, Insu et al. (2019). "Heterogeneous Isolated Execution for Commodity GPUs". In: *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*. ASPLOS '19. Providence, RI, USA: Association for Computing Machinery, pp. 455–468. ISBN: 9781450362405. DOI: 10.1145/3297858.3304021. URL: https://doi.org/10.1145/3297858.3304021.

Jang, Yeongjin et al. (2017). "SGX-Bomb: Locking Down the Processor via Rowhammer Attack". In: *Proceedings of the 2nd Workshop on System Software for Trusted Execution*. SysTEX'17.

Shanghai, China: Association for Computing Machinery. ISBN: 9781450350976. DOI: 10.1145/3152701.3152709. URL: https://doi.org/10.1145/3152701.3152709.

Knott, Brian et al. (2021). "CrypTen: Secure Multi-Party Computation Meets Machine Learning". In: *Advances in Neural Information Processing Systems*. Ed. by M. Ranzato et al. Vol. 34. Curran Associates, Inc., pp. 4961–4973. URL: https://proceedings.neurips.cc/paper/2021/file/2754518221cfbc8d25c13a06a4cb8421-Paper.pdf.

NVIDIA (2022). *NVIDIA H100 Tensor Core GPU Architecture: Exceptional Performance, Scalability, and Security for the Data Center*. Tech. rep.

Tramèr, Florian and Dan Boneh (2018). *Slalom: Fast, Verifiable and Private Execution of Neural Networks in Trusted Hardware*. DOI: 10.48550/ARXIV.1806.03287. URL: https://arxiv.org/abs/1806.03287.

Volos, Stavros, Kapil Vaswani, and Rodrigo Bruno (Oct. 2018). "Graviton: Trusted Execution Environments on GPUs". In: *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*. Carlsbad, CA: USENIX Association, pp. 681–696. ISBN: 978-1-939133-08-3. URL: https://www.usenix.org/conference/osdi18/presentation/volos.

Wagh, Sameer et al. (2020). *FALCON: Honest-Majority Maliciously Secure Framework for Private Deep Learning*. DOI: 10.48550/ARXIV.2004.02229. URL: https://arxiv.org/abs/2004.02229.

Wu, Chenwei, Chenzhuang Du, and Yang Yuan (2020). *Secure Data Sharing With Flow Model*. DOI: 10.48550/ARXIV.2009.11762. URL: https://arxiv.org/abs/2009.11762.

Xue, Mingfu et al. (2020). "Machine Learning Security: Threats, Countermeasures, and Evaluations". In: *IEEE Access* 8, pp. 74720–74742. DOI: 10.1109/ACCESS.2020.2987435.

BIOGRAPHICAL SKETCH

Lucas Dominic Hall was born September 19, 1981, in San Antonio Texas to his mother Melanie Zygmont. He and his older brother were both adopted at a young age by their father George Jeffrey Hall, and they were raised alongside their younger sister in the small rural town of Taylor Texas.

Mr. Hall first earned an Associate's Degree in General Studies from Temple College in Temple Texas in May of 2004. Later, he earned a Bachelor's Degree in Biology from University of Texas at Brownsville in December of 2008, and then earned a Bachelor's Degree in Computer Science in December of 2019 from University of Texas Rio Grande Valley. Finally, he has earned a Master's Degree in Computer Science in May of 2022 from University of Texas Rio Grande Valley.

Mr. Hall has held positions at many companies primarily related to information technology and other computer related technologies. Companies he has worked for in recent years include AT&T as well as Blizzard Entertainment. Mr. Hall is also a U.S. Navy Veteran with an Honorable Discharge. Mr. Hall can be contacted by reaching out to him at his longstanding email of lucashall1981@gmail.com.