ENGINEERING DESIGN 1
GROUP 8

Assignment 1: Project Summary

Dr. Zhong

Table of Contents

Project Sponsor Information

Marc Asselin – CEO of Ava Intelligence

Contact Information: marc@avaintell.com

Team Members

Sienna Gaita-Monjaraz – Computer Science

Contact Information: sgaitamonjar2019@fau.edu

Carlo Leiva – Computer Science

Contact Information: leivac2017@fau.edu

Tania Alam – Computer Science

Contact Information: talam2021@fau.edu

Aleysha Sierra Santiago – Computer Engineering

Contact Information: asierrasanti2022@fau.edu

Dien Tran – Computer Science

Contact Information: dientran2018@fau.edu

Project Overview

Homomorphic encryption (HE) allows cloud servers to operate on encrypted text without having to decrypt it. It is the process of applying computations directly on encrypted text, also known as cipher text, to create a new encrypted text. The result is the same as the result of performing the operations on the decrypted data, known as plaintext. The owner receives the result and the privacy of the data is preserved. There are three levels of homomorphic encryption.

- *Fully homomorphic encryption* is robust as it can use any number of operations which is why it's the most researched and popular.
- *Somewhat homomorphic encryption* uses two operations, but the operations are limited.
- *Partially homomorphic encryption* uses either the addition or multiplication operation. Though this kind of encryption has limitations, it is easy to design.

However, the computational overhead associated with HE has been a significant deterrent to its practical implementation. This is why innovative solutions like this are being pushed to mitigate this drawback. We want to harness the potential of HE for practical use, especially in machine learning applications, by exploring hardware acceleration and optimizing encryption techniques to address the computational challenges inherent in HE.

The objective of this project is to develop a library for homomorphic encryption utilizing Python. By doing so, it is envisioned that this could mitigate the computational overhead of

homomorphic encryption, foster secure data processing in machine learning applications, and a viable hardware-software synergy to accelerate encrypted data processing. As a solution, a key focus of this project is to use hardware, such as FPGA's, to speed up the performance of computationally intensive tasks.

This project will have three key pillars to it: homomorphic encryption library development, textual data encryption, and hardware acceleration. This will extend to scrutinize existing encryption libraries, design a library architecture, and integrate hardware acceleration techniques to optimize the performance of encrypted data processing. However, it's important to note that the scope will not go into the broader application of the developed library in external systems.

Success in this project is envisioned as a robust, user-friendly Python library or API capable of efficient encrypted data processing. The deliverables delineated for this project are a fully documented Python library or API, a demonstration of the library's efficacy in a controlled environment, and performance testing reports that elucidate the computational advantages garnered through hardware acceleration.

This expected timeline of this project consists of 18 weeks broken down into 2-3 weeks for each phase, as seen below.

- Week 1-2: Research Phase
- Week 3-4: Design Phase
- Week 5-9: Development Phase
- Week 10-12: Testing & Integration Phase
- Week 13-14: Documentation & Demonstration Phase

- Week 15-16: Feedback & Iteration Phase
- Week 17-18: Preparation for Showcase

Project Scope

Homomorphic Encryption

This project focuses on providing a comprehensive overview of Homomorphic Encryption, a cryptographic method enabling computation on encrypted data. The team will explore libraries like Microsoft's SEAL and PALISADE and clarify distinctions between types of homomorphic encryption (Partially, Somewhat, and Fully). Two individual team members will be assigned to this task.

Textual Data Encryption Techniques

In addition to homomorphic encryption, the project explores efficient encryption methods for textual data. This includes symmetric key Encryption (like AES) for bulk textual data encryption, public key infrastructure (PKI) for secure key exchange and Secure multi-party computation (SMPC) for multiple party scenarios. Two individual team members will be assigned to this task.

Hardware Acceleration and Performance Enhancement

The project evaluates hardware acceleration for improving encryption task performance. This entails examining GPU acceleration (CUDA), FPGA tools for custom acceleration, and studying prior work on hardware-accelerated homomorphic encryption for guidance. One individual team member will be assigned to this task.

Deliverable

The Project's key deliverables are the development of a python library for Homomorphic Encryption with documentation and usage examples, a practical demonstration of the library in action, an analysis of library performance with and without hardware acceleration, and a projection presentation summarizing findings. The process will also involve collection feedback and making iterative inputs based on inputs.

Key Challenges

Homomorphic Encryption:

- Complexity and Computational Overhead: This kind of encryption is computationally intensive, it can significantly slow down operations. Overhead can be a substanstial challenge for real-time or performance-critical applications, and it can leads to increased response times.
- Limited Practical Applications: Due to the computational overhead and complexity, we have seen limited use in real-world applications. Navigating through research papers is a dense and time consuming task especially when there is limited availability because of resistried access or papers only accessible to certain research institutions. Homogrpahic encryption is most beneficial in situations where security is the most important and performance impact is acceptable.
- Understanding Homomorphic Encryption Types: There are several types of homomorphic encryption including fully homomorphic (allows arbitrary computations), partially homomorphic (allows only one type of operation on ciphertexts), and somewhat homomorphic

(allow limited combinations of operations). Understanding these is a crucial part of the process. Choosing the right type for our case is a difficult task as it depends on the desired operations and performance constraints.

Textual Data Encryption:

- Overhead and Inefficiency: Textual data like plain text documents can expand in size when encrypted. This expansion can make data storage and transmission less efficient. While trying to balance the need for security with handling large, encrypted textual datasets it can become complicated.
- The Right Encryption Technique: The challenge here lies with choosing the correct technique, symmetric key encryption like the advanced encryption standard (AES) is ideal for fast bulk encryption of textual data. This method may not offer the same level of security as homomorphic encryption. We are going to have to take careful consideration of specific requirements when making a decision.
- Secure Key Exchange: When working with textual data encryption ensuring secure key exchange is crucial. Public Key Infrastructure (PKI) is used to establish source communication channels, but protecting encryption keys will be the ongoing challenge. Key management must address issues like key distribution, rotation and cancellation while still keeping sensitive data safe.

Hardware Acceleration:

- Compatibility and Integration: Integrating hardware acceleration solutions like GPUs and FPGAs into a system can be complex. Ensuring that these solutions are compatible with the software and infrastructure plays a critical role in enabling efficient implementation. Incompatibles can lead to performance limitations and even system failures.

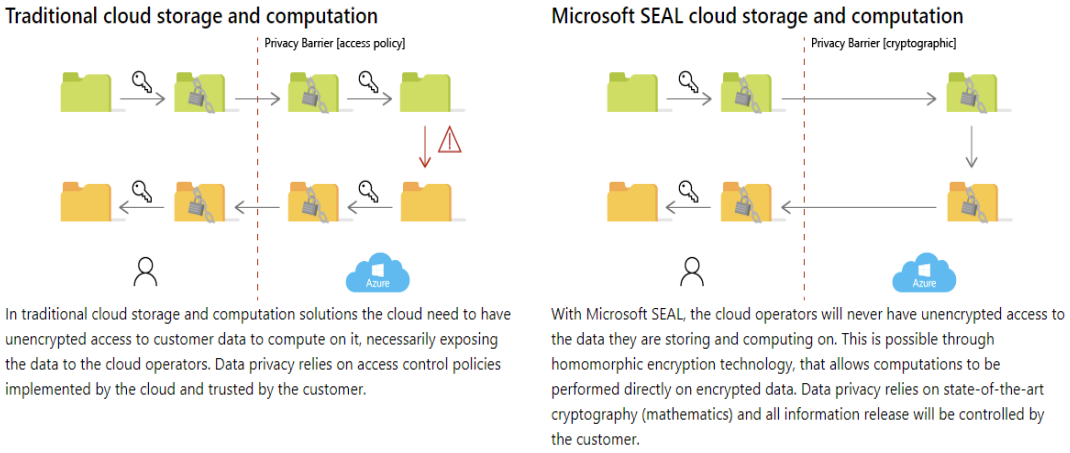
- Optimizing CUDA for GPU Acceleration: GPUs can offer significant speed-ups for certain computations, being able to optimize algorithms and code to fully leverage GPU capabilities can be rigorous and time-consuming. We need to design and implement a parallel algorithm in some capacity that can make full use of the GPUs processing power, memory, and threading capabilities.
- FPGA Development and Customization: FPGA development tools provide a unique advantage for customization, it allows us to tailor the hardware to fit any specific task. To do this we would need a deep understanding of FPGA programming languages such as VHDL or Verilog and without prior knowledge of this the task can seem daunting. An alternative solution would be to use High-Level Synthesis tools that could help write FPGA accelerators by converting python code to FPGA configurations indirectly. This approach can significantly help with the learning curve that comes with FPGA development.

Key Technologies

Homomorphic Encryption: A class of encryption methods first constructed in 2009 but it was an idea that was dated back to 1978. This type of encryption allows computation to be performed directly on encrypted data without requiring access to a secret key. Thus, the computation remains in encrypted form, and can be revealed later by the owner of the secret key. A reason why Homomorphic Encryption is so desired is because of how simple it is, a cloud server can directly operate on the encrypted data and return only the encrypted result to the owner of the data, unlike traditional encryption methods, such as AES, which require the cloud server needing access to the secret key or having the owner of the data download, decrypt and operate on it locally. How this method of

encryption works is based on the hard mathematical problem related to high-dimensional lattices: Ring-Learning with Errors (RLWE). This RLWE problem is secured against quantum computers.

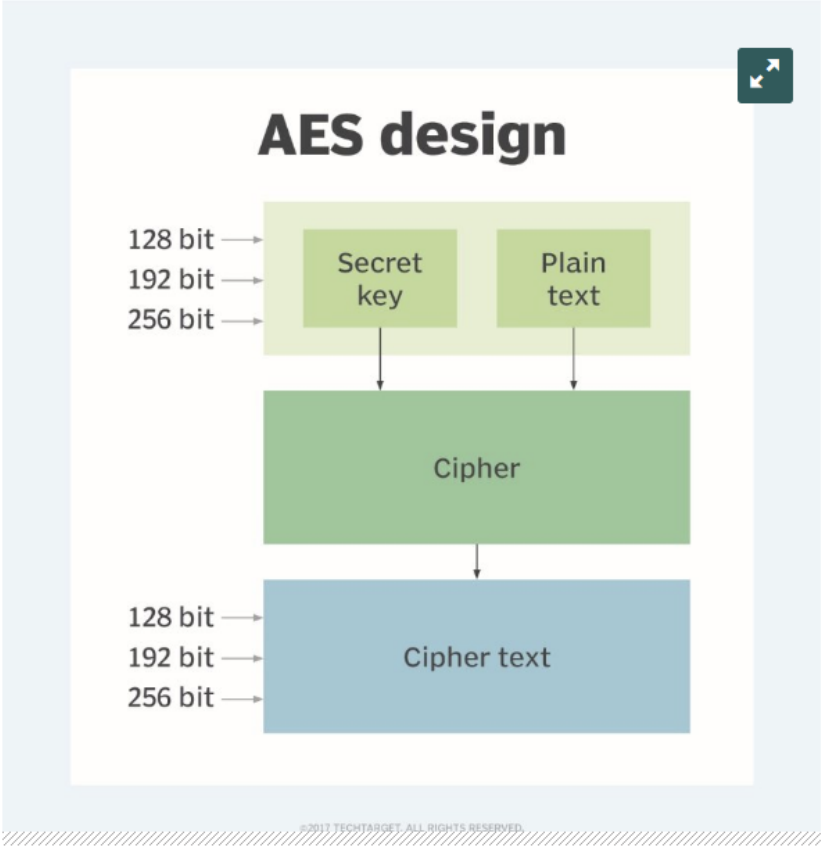
- Microsoft's SEAL – Microsoft provides a simple, yet effective set of open-source encryption libraries in the form of an API (Application Programming Interface) that allows computation to be performed directly on the encrypted data. This allows software engineers to directly build end-to-end encrypted data storage and computation services where customers never share their key with the service.



- Palisade – Palisade is an open-source project that provides simple APIs (Application Programming Interfaces) for efficient implementation of lattice cryptography building blocks and leading homomorphic encryption schemes. Palisade is designed for usability, modularity, cross-platform support, and integration of hardware accelerators. Palisade supports the BGV, BFV, CKKS, and FHEW schemes and a more secure variant of the TFHE Scheme, including bootstrapping. Along with this, the API also provides post-quantum public-key encryption, proxy re-encryption, threshold FHE for multiparty computations, identify-based encryption, attribute-based encryption and digital signature support.

Textual Data Encryption: Encryption is the process of translating plain text data into something that is random and meaningless, which is ciphertext. The goal of this process is to make it as difficult as possible to read and decrypt the generated ciphertext without using the key. If a good encryption algorithm is used, then there is no technique significantly better than methodically trying every single key. The longer the key, the more difficult and more secure the ciphertext is.

- AES (Fast, bulk encryption) - Advanced Encryption Standard or AES for short is a symmetric block cipher used by the U.S. Government to protect and secure classified information. AES is implemented in hardware and software throughout the world to encrypt sensitive data for computer security, cybersecurity, and electronic data protection.
    - AES works by three different block ciphers:
        - AES-128, which uses a 128-bit key length to encrypt and decrypt a block of message
        - AES-192, uses 192 respectively.
        - AES-256, which uses 256
    - Each cipher is encrypted and decrypted in blocks using cryptographic keys of 128, 192, and 256 respectively. The secret key or Symmetric is something both the sender and receiver must know to encrypt and decrypt. The government uses this method to classify information into three different categories: Confidential, Secret, or Top Secret.



- PKI (Secure Key Exchange) - Public key infrastructure or PKI is the most common form of encryption, which uses a public key that allows anyone to encrypt a message, however, for the message to be decrypted, the user must have the private key or secret key.
    - PKI works by either using Symmetric Encryption or Asymmetric Encryption.
- SMPC (Multiple parties) - Secure Multiparty Computation or SMPC is a cryptographic protocol that allows multiple parties involved to compute on distributed data without ever having the need to expose it or move it. Neither party has access to each other's data.

Hardware Acceleration:

- CUDA for GPU acceleration – CUDA is a parallel computing platform and programming model invented by NVIDIA. It enables dramatic increases in computing performance by harnessing

the power of the GPU (Graphic processing unit). CUDA provides a small set of extensions for C/C++ languages that allows for straightforward implementation of parallel algorithms. Users can spend more. Time on the task of parallelization rather than implementation.
CUDA-capable GPUs have hundreds of cores that can collectively run thousands of computing threads allowing for shared resources including a register file and a shared memory. The on-chip shared memory allows parallel tasks running on these cores to share data without sending it over the system memory bus.

- System requirements
    - A CUDA-capable GPU
    - A supported version of Linux with a GCC compiler and toolchain
    - NVIDIA CUDA Toolkit

Python: Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Python is simple and is easy to read. It has high-level built in data structures, combined with dynamic typing and dynamic binding, making it very attractive for rapid application development, scripting or the glue language to connect existing components together.

References

Bernstein, C., & Cobb, M. (2021, Sept.). *What is the Advanced Encryption Standard (AES)? Definition from SearchSecurity*. TechTarget. Retrieved October 22, 2023, from https://www.techtarget.com/searchsecurity/definition/Advanced-Encryption-Standard

CUDO. (2023, Oct. 10). *1. Introduction — Installation Guide Windows 12.3 documentation*. NVIDIA Docs. Retrieved October 22, 2023, from https://docs.nvidia.com/cuda/cuda-installation-guide-microsoft-windows/index.html

Edgeless Systems. (n.d.). *The differences between Homomorphic Encryption(HE) and Confidential Computing (CC)*. Edgeless Systems. Retrieved October 22, 2023, from https://www.edgeless.systems/blog/the-differences-between-homomorphic-encryption-he-and-confidential-computing-cc/

Inpher. (n.d.). *What is Secure Multiparty Computation? - SMPC/MPC Explained*. Inpher. Retrieved October 22, 2023, from https://inpher.io/technology/what-is-secure-multiparty-computation/

Introduction – Homomorphic Encryption Standardization. (n.d.). Homomorphic Encryption Standardization. Retrieved October 22, 2023, from https://homomorphicencryption.org/introduction/

Keyfactor. (n.d.). *What is PKI? A Public Key Infrastructure Definitive Guide*. Keyfactor. Retrieved October 22, 2023, from https://www.keyfactor.com/education-center/what-is-pki/

Microsoft. (n.d.). *Microsoft SEAL: Fast and Easy-to-Use Homomorphic Encryption Library*. Microsoft. Retrieved October 22, 2023, from https://www.microsoft.com/en-us/research/project/microsoft-seal/

Microsoft. (2022, February 7). *Data encryption and decryption - Win32 apps*. Microsoft Learn. Retrieved October 22, 2023, from https://learn.microsoft.com/en-us/windows/win32/seccrypto/data-encryption-and-decryption

PALISADE Homomorphic Encryption Software Library – An Open-Source Lattice Crypto Software Library. (n.d.). Retrieved October 22, 2023, from https://palisade-crypto.org/

Programming an FPGA: An Introduction to How It Works. (n.d.). AMD. Retrieved October 22, 2023, from https://www.xilinx.com/products/silicon-devices/resources/programming-an-fpga-an-introduction-to-how-it-works.html

Python. (n.d.). *What is Python? Executive Summary*. Python.org. Retrieved October 22, 2023, from https://www.python.org/doc/essays/blurb/