ÇANKAYA UNIVERSITY
FACULTY OF ENGINEERING
COMPUTER ENGINEERING DEPARTMENT

Project Report

Version 1

CENG 408

Innovative System Design and Development I

201912

A CUDA Based Disk and File Encryption

Fatih Taşdemir
201511058
Muhammed Bera Erkaya
201511022
Emre Yüncü
201511068
Doğa Üçüncü
201411063

Advisor: *Asist. Prof. Dr. Hüseyin Temuçin*

Table of Contents

Abstract

The files on the computer system are kept in plain form, and third parties who access the disk via physical or network can read this data. Within the scope of the project, a file encryption library will be developed and AES algorithm CTR mode will enable encryption / decryption of encrypted files. Since the files are large-scale data, CUDA will be used to make this process faster and encryption will be performed on the GPU.

Key words:

File encryption, CUDA, encryption, decryption, block encryption, AES algorithm, CTR Mode, CPU, GPU.

Özet:

Bilgisayar sistemindeki dosyalar düz biçimde tutulur ve diske fiziksel veya ağ üzerinden erişen üçüncü taraflar bu verileri okuyabilir. Proje kapsamında bir dosya şifreleme kütüphanesi geliştirilecek ve AES algoritması CTR modu, şifrelenmiş dosyaların şifrelenmesini / şifresinin çözülmesini sağlayacaktır. Dosyalar büyük ölçekli veriler olduğundan, bu işlemi daha hızlı hale getirmek için CUDA kullanılacak ve GPU'da şifreleme gerçekleştirilecektir.

Anahtar Kelimeler:

Dosya şifreleme, CUDA, şifreleme, şifre çözme, blok şifreleme, AES algoritması, Sataç modu CPU, GPU.

Introduction

Nowadays, many files are shared on the internet. They must be protected and not in the hands of third parties. Our project aims to protect the data by encrypting large-scale files. Because our files are large, we use Cuda to achieve our goal faster.

CUDA is a parallel computing platform and a programming model that makes GPU usage for general purpose computing simple and neat. The developers still work on C, C++, Fortran, and expand the list of supported languages. Moreover, they incorporate extensions of these languages in the form of a few basic keywords.

To encrypt, symmetric encryption will be applied. Symmetric encryption is the most preferred encryption model today. Because symmetric encryption is hard to break it. But for users it is easy to use. As an algorithm, we will use the CTR mode of the AES algorithm. AES algorithm is one of most using algorithm of symmetric encryption. Because it is more safe than other algorithms. The main reason we use CTR mode is that it enables encryption to be parallel in this mode.

We are developing CUDA based file encryption tool. Users can encrypt their files when they add their own files to this tool. With CUDA encryption will be faster and with symmetric encryption it will be safer.

Scope of Project

With this tool people can encrypt their files. Because we use CUDA for GPU's CPU. And it work faster than pc's cpu. And this tool compatible with C/C++. We will use symmetric encryption. And We use AES algorithm CTR mode for encryption. AES (Advanced Encryption Standard) is a standard for encryption of electronic data. AES, adopted by the US government, is also used internationally encryption (crypto) standard. The encryption algorithm defined by AES is a symmetric-key algorithm in which the keys used for both encryption and decryption are related. The encryption and decryption keys for AES are the same. CTR mode, on the other hand, will enable us to create a parallel structure for large files with initialization vector created by the counter.

1. Literature Review

1.1 Project Related Sections

Our work in our project is divided into two important sections. One is Cuda and the other is Encryption.

1.1.1 CUDA

- The main task of the Graphics Processing Unit (GPU) is to display the images generated on the computer.
- Due to the insufficient CPU computing problems, the parallel structure of the GPU has begun to be utilized.
- CUDA is a parallel computing architecture introduced by NVIDIA in 2006 to take advantage of the computing power of the GPU.
- It can run on Linux, Windows and Mac Osx platforms.
- CUDA can support programming languages such as C, C ++, Python.
- The CUDA interface allows faster data readings than the GPU compared to the CPU.

  GPU differs from CPU in that it has a Single Instruction Multiple Data (SIMD) architecture. CPU calculations are performed in series. GPU calculations are performed in parallel.

1.1.2 CUDA Work Flow

- Applications developed in CUDA architecture do not only work on the GPU. First, it must be copied to the memory on the graphics card via the main memory controlled by the CPU.
- The data in the GPU memory is executed by the CUDA threads and the calculation is completed in parallel. Then sent back to the main memory to finish the process.
- The pieces of code running on the CPU are different from the pieces of code running on the GPU.

The "Host" can be considered a CPU. The "Device" can be considered a GPU. Serial codes are executed on the CPU. Parallel pieces of code called "kernel" are executed on the GPU.

1.1.2.1 CUDA Programming Architecture

- Kernel
  - The part of a code developed in CUDA that will run on the GPU side is called "kernel".
  - GPU creates a kernel copy for each element of the dataset and is called a "thread".
  - The kernel code is invoked by the Host and executed on the Device. Kernel code is considered "global".
- Thread
  - Thread is the smallest thing in CUDA architecture. In blocks, they can be 1D, 2D or 3D.
  - They run the same piece of code simultaneously.
  - The threads are arranged in blocks and grouped. Threads in different Blocks do not work together.
  - Each thread has its own ID in the block. These indices; "threadIdx.x", "threadIdx.y" and "threadIdx.z".
- Block
  - "Block" structure consists of parallel threads. They are unique in the "grid". They can be 1D, 2D or 3D within the grid.
  - The blocks are arranged in grid and grouped.. Each Block has its own index in the Grid. These indices are "blockIdx.x", "blockIdx.y" and "blockIdx.z".
  - They are dimensioned according to the number of rows and coulumns, such as “blockDim.x”, “blockDim.y”, blockDim.z”.
- Grid
  - Grid is a structure that "blocks" come together. Each "kernel" call creates a "grid".
  - Grid dimensions can be expressed as “gridDim.x”, “gridDim.y” and “gridDim.z”. [1]

1.2.2 Encryption

Encryption is an encryption process to prevent data from being read by people and other computers and to prevent the original content from being accessed.

Each Encrypt operation is performed according to a specific algorithm and the encrypted data can be made readable with a simple solver. Encrypting data appears to be pointless until decrypted and is nonfunctional.

1.2.2.1 Symmetric Encryption

Symmetric encryption is an encryption process in which the same key is used to both encrypt and decrypt data. This type of information coding method has been used frequently in the past decade to ensure confidential communication between states and armies. Nowadays, symmetric encryption algorithms are widely applied to improve data security in various computer systems.

Symmetric encryption process are based on only one key shared by two or more users. The same key is used to encrypt and decrypt for plain text. The encryption process consists of creating a ciphertext by passing a plaintext through an encryption algorithm called cipher.[2]

If the encryption process is strong enough, there is only one way for people to access and read information in the ciphertext. Using the key to decrypt it. The decryption process consists essentially of converting the encrypted text back to plain text.

The security of symmetric encryption is based on how difficult it is to estimate the key corresponding to the system by applying brute force. For example, it takes billions of years to estimate a 128-bit key using ordinary computer hardware. The longer the password key, the harder it is to break it. 256-bit keys are generally considered very secure and theoretically resistant to brute force attacks by quantum computers. Symmetric encryption has advantages over itself. Encryption and decryption are quick, easy to implement with hardware. Confidentiality of communication between the parties is ensured. The integrity of the data is ensured. The original text cannot be changed unless the encrypted text is decoded. [3]

1.2.2.2 AES Encryption Algorithm

The more popular and widely adopted symmetric encryption algorithm likely to be encountered nowadays is the Advanced Encryption Standard (AES). It is found at least six time faster than triple DES.

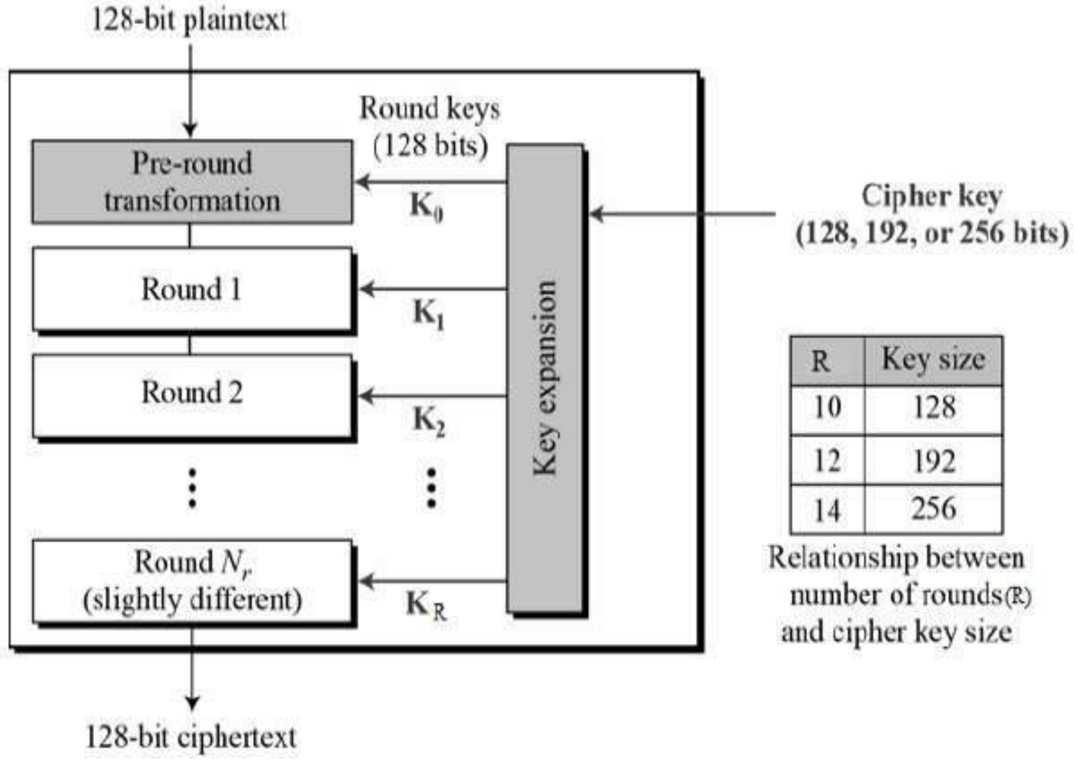
A replacement for DES was needed as its key size was too small. With increasing computing power, it was considered vulnerable against exhaustive key search attack. Triple DES was designed to overcome this drawback but it was found slow.[4]

1.2.2.2.1 Operation of AES

AES is a symmetric encryption algorithm, that is, the same key is used for both encryption and decryption. The input and output matrices for AES must always be 128 bits, but the key length can be 128, 192 or 256 bits.

The matrix consists of 4 rows, 4 columns, 16 divisions. Each partition occupies 8 bits of space, as there are 16 partitions corresponding to a total of 128 bits of data. Necessary actions are taken on them by thinking as messages and key status matrices to be encrypted.

The AES algorithm can be considered as repetition of steps such as Add round Key, Substitute bytes, Shift rows, and Mix columns. The encrypted message is reached after 10 rounds for 128 bit AES encryption, 12 for 192 bit and 14 rounds for 256.[5]

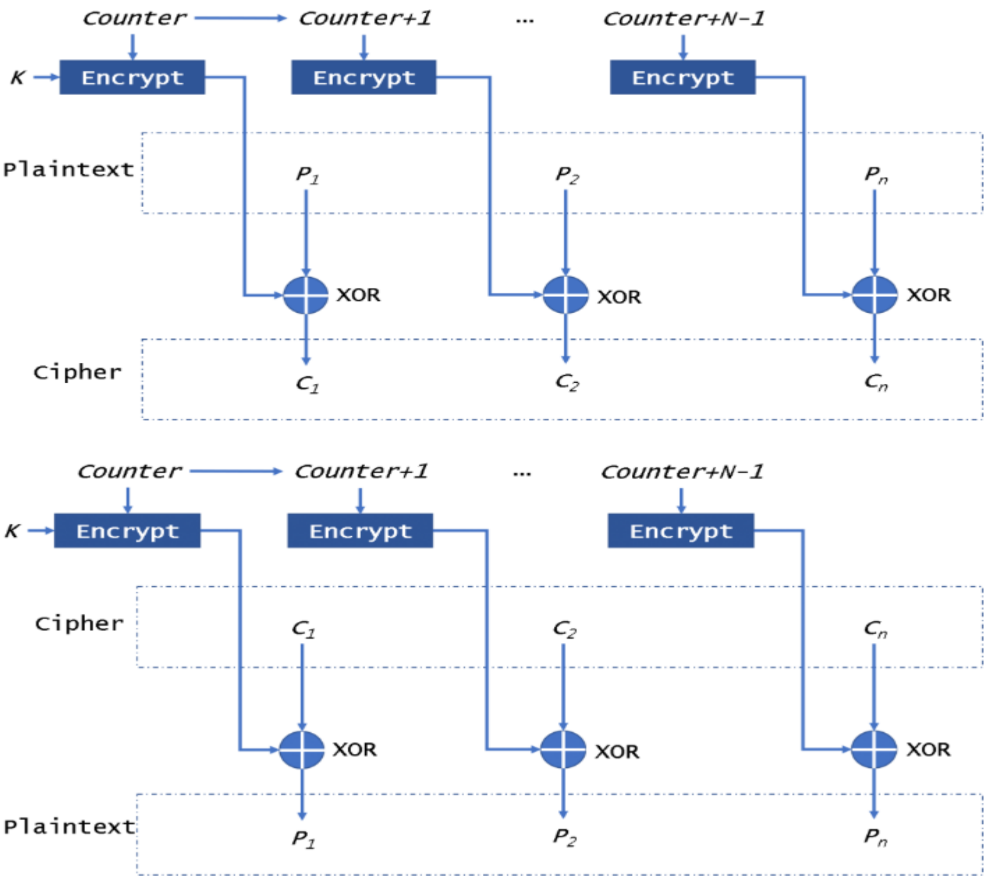


R	Key size
10	128
12	192
14	256

Relationship between number of rounds(R) and cipher key size

1.2.2.2.2 CTR MODE

The counter has the same size as the used block. The XOR operation with the block of plain text is performed on the output block from the encryptor. All encryption blocks use the same encryption key. As this mode, It will not be affected by the broken block. CTR will use the counter to be encrypted every time instead of the IV. So if you could get counter directly, you can encrypt/decrypt data in parallel.[6]



2. Software Requirements Specification

2.1 Purpose

The files on the computer system are kept in plain form, and third parties who access the disk via physical or network can read this data. Within the scope of the project, a file encryption library will be developed and AES algorithm CTR mode will enable encryption / decryption of encrypted files. Since the files are large-scale data, CUDA will be used to make this process faster and encryption will be performed on the GPU.

2.1.1 Glossary

- CUDA- Compute Unified Device Architecture
- AES- Advanced Encryption Standard(Encryption algorithm)
- ENC-Encryption
- DEC- Decrption
- CPU- Central Processing Unit
- GPU- Graphics Processing Unit
- CTR Mode - Counter Mode

2.2 Overall Description

The SRS document will include performance and interface requirements, operations and functions, software and system features, and interface information for this project.

2.2.1 Product Perspective

This product will be tool. Users can use this for encrypt/decrypt files. This system work with CUDA because this system use GPU’s CPU.

2.2.2 Operations

The user should select a file using the menu provided to him. The selected file is presented as output. If the file can be read, it can encrypt. If it is unreadable, it can be decrypted. Since we use AES encryption CTR mode, our encryption process is both block and parallel. We check the file sizes. If the file size is large, we use CUDA programming. CUDA provides us with the thread structure faster encryption. We use a thread for each block. If the file is too large, it can be divided into 256-bit sizes. In this way, it can work faster. Each result is printed to an output file.

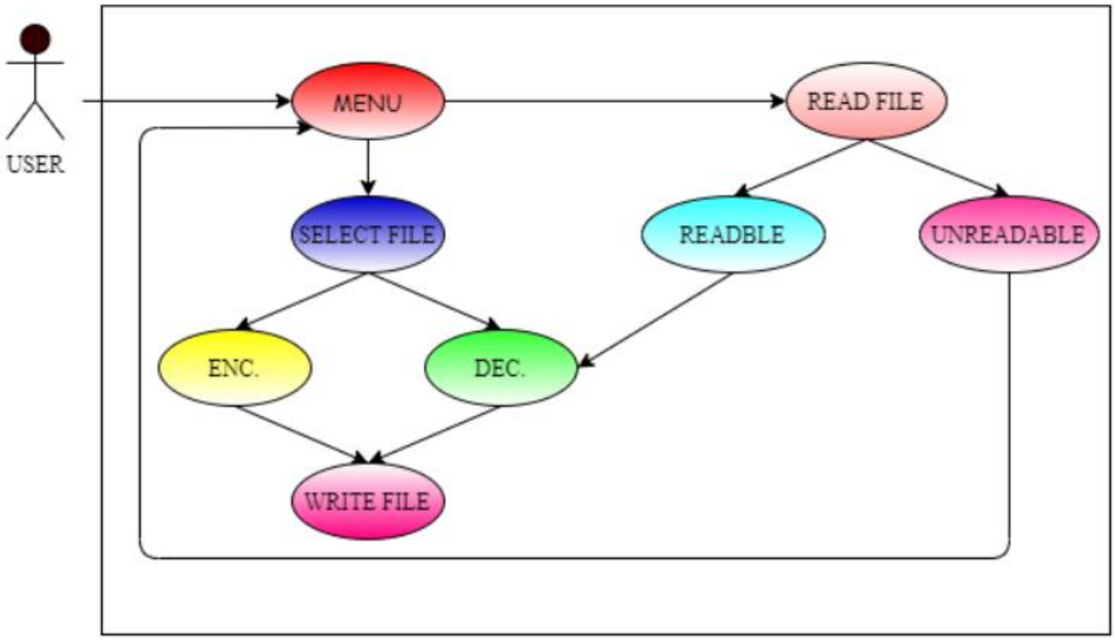
2.2.3 System Adaption Requirements

In our project, we will open the file with the help of the menu and convert the text into encrypted text with AES encryption CTR mode. Since we use AES CTR mode, all blocks to be encrypted have a counter number.In this way, files can be both easily encrypted and decrypted.We can do this in 256, 512 or 1024 blocks. We will determine the efficiency of the encryption process in a direct proportion. Our library encrypt a file which you want.When user add this library, the system encrypts the file with CUDA using symmetric encryption using CPU of the GPU.Users have a key for encrypt and decrypt. User select file for encryption. Then with our library system encrypt this file automatically. When user want decrypt this file users need their symmetric key. Otherwise users can not access their file.

2.2.4 Product Funcitons

In this project, we can split the file to be encrypted into 256, 512 or 1024 blocks and perform the encryption process gradually. We will decide the number of blocks to be directly proportional to the efficiency we will receive in encryption operations. With this library, there will be encryption and decryption on file. The other function of this library is in system “a” it’s encrypted and then in system b it is decrypted. At first system “a" encrypt their file and system “a” has key for encryption. Then in system “b” has a key for decryption. Without this key system “b” can not decrypt this file. These keys can be128,192 or 256-bit long. User try to decrypt file.If the key is wrong, the file will be corrupted and it will be more difficult for 3rd parties to solve the file.This work with symmetric encryption. We use AES algorithm, because nowadays this algorithm is more safe than other algorithms.We will also use the AES CTR mode because we are trying to create a parallel structure.This symmetric encryption works compatible with CUDA.

2.2.5 Use Case Diagram



2.3 Specification of Requirements

2.3.1.1 Software Interfaces

Anyone using the Linux operating system and NVidia GPU can use this project. Our project is written in C and CUDA programming language.

2.3.1.2 Performance Requirements

Since we will run our project over GPU, the number of cores in the GPU will be important for us. The NVIDIA graphics card is critical, as we will also use the CUDA language. In NVIDIA graphics cards, the number of cores increases as the models increase. For these reasons, the NVIDIA graphics card models and the number of cores are important elements for our project. We plan to use NVIDIA GEFORCE GTX 1050 TI and higher models. On the encryption side, we choose symmetric encryption and the CTR mode of the AES algorithm. We choose the AES algorithm because it is one of the safest algorithms in the world and allows us to build a parallel structure.

2.3.2 Software System Attributes

2.3.2.1 Portability

At the end of the project we will add our own file structure. This file structure will have a structure that has its own header like pdf. This file structure will add mobility to us. In this way, we will provide greatly convenience in running our file.

2.3.2.2 Performance

Since we will use the AES algorithm on the encryption side, it will be more secure. AES algorithm already included in many encryption packages is the first publicly available encryption algorithm approved by the NSA(National Security Agency) to encrypt confidential information. For a parallel structure, we will use the AES CTR mode. We use CUDA to make the system encrypt faster.

2.3.2.3 Adaptability

Our library will be compiled and executed on any linux operating system. Since the algorithm we will use for encryption will be specific to CUDA, we will need a GPU with CUDA support. If the GPU is not supported by CUDA, the algorithm will be executed through the CPU and the desired efficiency will not be achieved. We also plan to create this algorithm and library using C programming languages.

2.3.2.4 Safety Requirements

This is encryption tool. Most important thing is you must not loose your key and don’t give your key to other people. We will use one of most secure encryption algorithm. There are 2128 different keys in AES 128-bit encryption, and it takes a great deal of time and cost to decrypt it. Suppose that a person uses all the technologies currently available as hardware to decode a 128-bit password, it takes approximately 100 years to crack the password. It means at 256-bit it is very long time. In our project, the key will be held with a separate file. If this file does not exist, the user cannot encrypt or decrypt the files. If decryption is applied with an incorrect key to read the file, the file structure may be corrupted completely. Care should be taken in this regard. For this reason, our system is protected against brute force attacks. [7]

3.Software Design Descriptions

3.1.1 Purpose

- Information and communication technologies are evolving and businesses are restructuring on the basis of new technology and they are trying to achieve more successful activities, services and products. There are also a number of problems associated with the fact that information is important to companies. These are the concerns about data security. A number of measures need to be taken to ensure that these concerns are eliminated. [8]

- Your data can be accessed at the border, taken from you in the street or from your house and copied in seconds. Unfortunately, locking your device with passwords can not protect your data if the device itself is seized. Other people would just need to access the storage directly in order to copy your data without your password.[9]

- If you use encryption for your data, other people need both your device and your password to unscramble the encrypted data. This way, it's safest to encrypt all of your data, not just a few data. For this, smartphones and computers provide complete full-disk encryption.

- While encrypt and decrypt operations are performed on small files, CPU power may be insufficient for larger files, while processing power may be insufficient in terms of speed. Therefore, the GPU can be used to speed up this processing time.

- CUDA technology allows parallel programming because the CPU is suitable for parallel programming due to its architecture. This parallelism allows us to perform many operations at the same time and gives us great speeds when encrypt and decrypt large files.

3.1.2 Glossary

- CUDA- Compute Unified Device Architecture
- AES- Advanced Encryption Standard(Encryption algorithm)
- ENC-Encryption
- DEC- Decrption
- CPU- Central Processing Unit
- GPU- Graphics Processing Unit
- CTR Mode – Counter Mode

3.1.3 Overview of Document

More detailed information of the rest of the content is clarified in the below sections. Section 2.1. is the "Design Approach" which contains information about the development methodology of the project.

Section 2.2 contains information about which technologies and which encryption algorithms our project will have and how to use them.

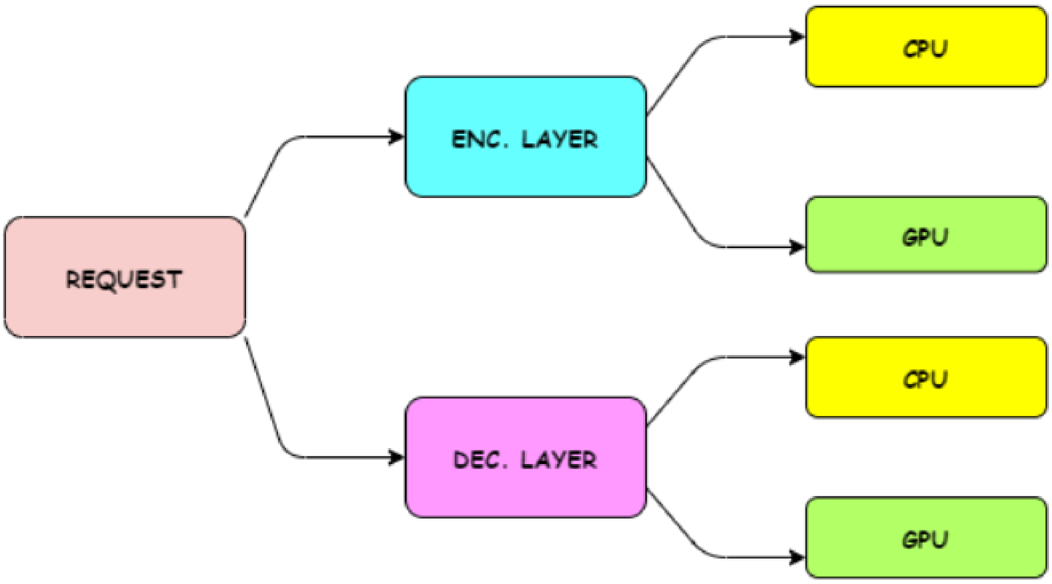
Section 2.3 contains the activity diagram of the project that represents the workflows of activities and actions.

3.2 Architecture Design

3.2.1.1 Design Approach

The aim of the project is to encrypt files fast and low cost. For this reason, our project we designed works as follows. The user should select a file using the menu provided to him. The selected file is presented as output. If the file can be readable, it can encrypted. If it is unreadable, it can be decrypted. Since we use AES encryption CTR mode, our encryption process is both block and parallel. We check the file sizes. If the file size is large, we use CUDA programming. If the file size is small, we use C programming. GPU performs better on large files. The CPU performs better on smaller files. CUDA allows us to access the GPU. C programming runs on the CPU. CUDA provides us with the thread structure faster encryption. We use a thread for each block. If the file is too large, it can be divided into 256-bit sizes. In this way, it can work faster. Each result is printed to an output file.

3.2.1.2 Encryption/ Decryption Engine Layer Design



3.2.1.3 Encryption Layer – AES Encryption CTR Mode

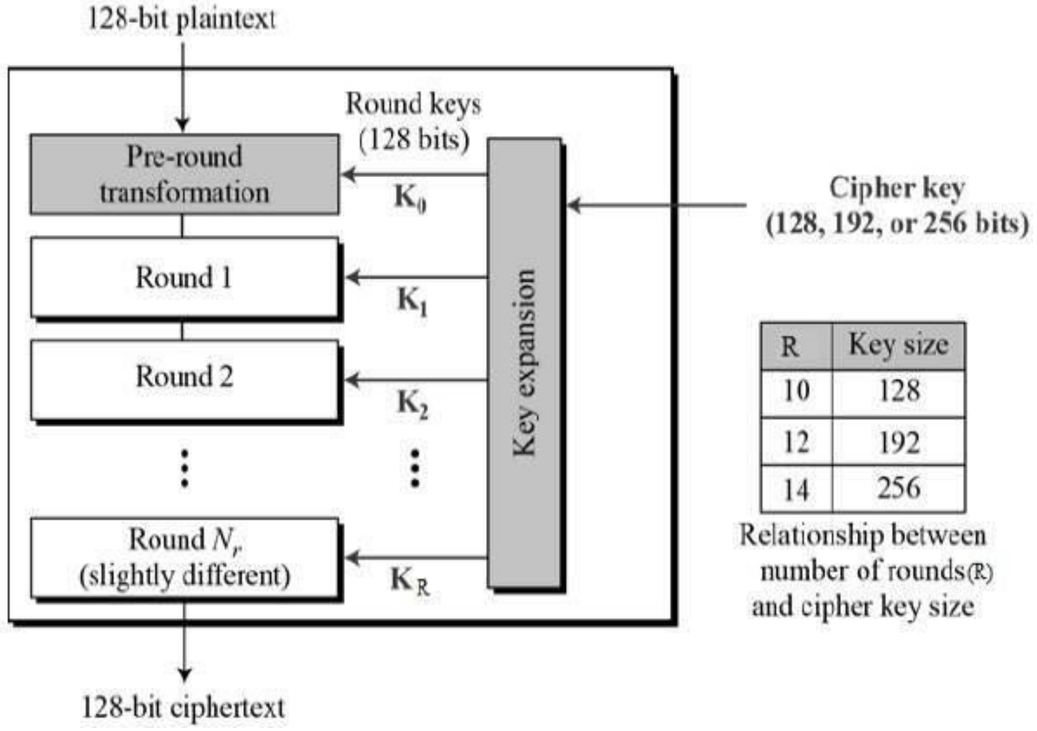
- AES is a symmetric encryption algorithm, that is, the same key is used for both encryption and decryption. The input and output matrices for AES must always be 128 bits, but the key length can be 128, 192 or 256 bits.

- The matrix consists of 4 rows, 4 columns, 16 divisions. Each partition occupies 8 bits of space, as there are 16 partitions corresponding to a total of 128 bits of data. Necessary actions are taken on them by thinking as messages and key status matrices to be encrypted.

- The AES algorithm can be considered as repetition of steps such as Add round Key, Substitute bytes, Shift rows, and Mix columns. The encrypted message is reached after 10 rounds for 128 bit AES encryption, 12 for 192 bit and 14 rounds for 256.[10]

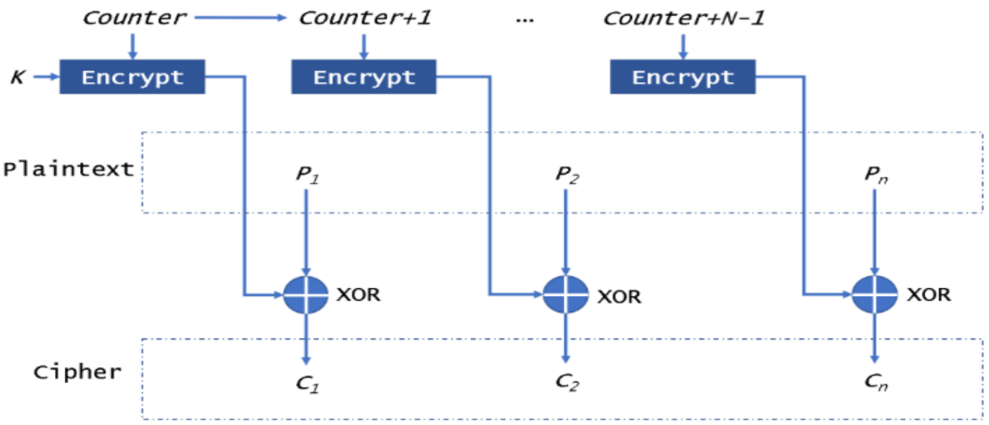
- The counter has the same size as the used block. The XOR operation with the block of plain text is performed on the output block from the encryptor. All encryption blocks use the same encryption key. As this mode, It will not be affected by the broken block. CTR will use the counter to be encrypted every time instead of the IV. So if you could get counter directly, you can encrypt/decrypt data in parallel.[11]

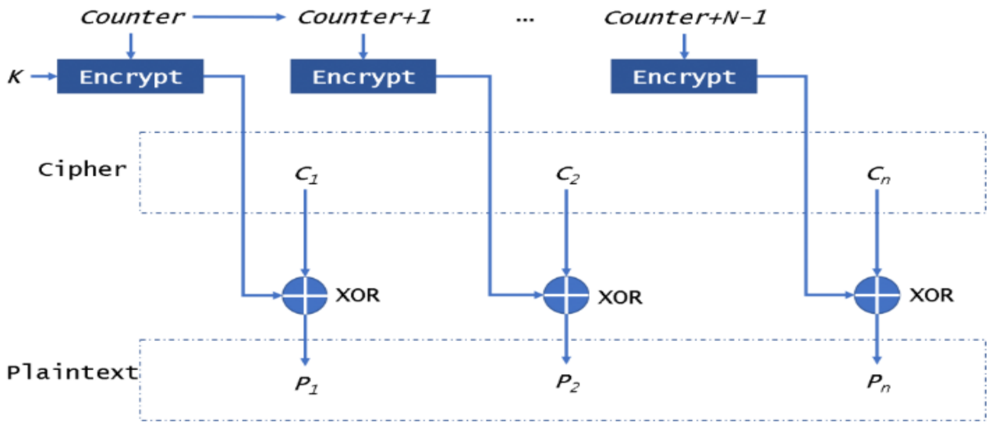
  You can see the AES and CTR mode layers from the figures below.



R	Key size
10	128
12	192
14	256

Relationship between number of rounds(R) and cipher key size

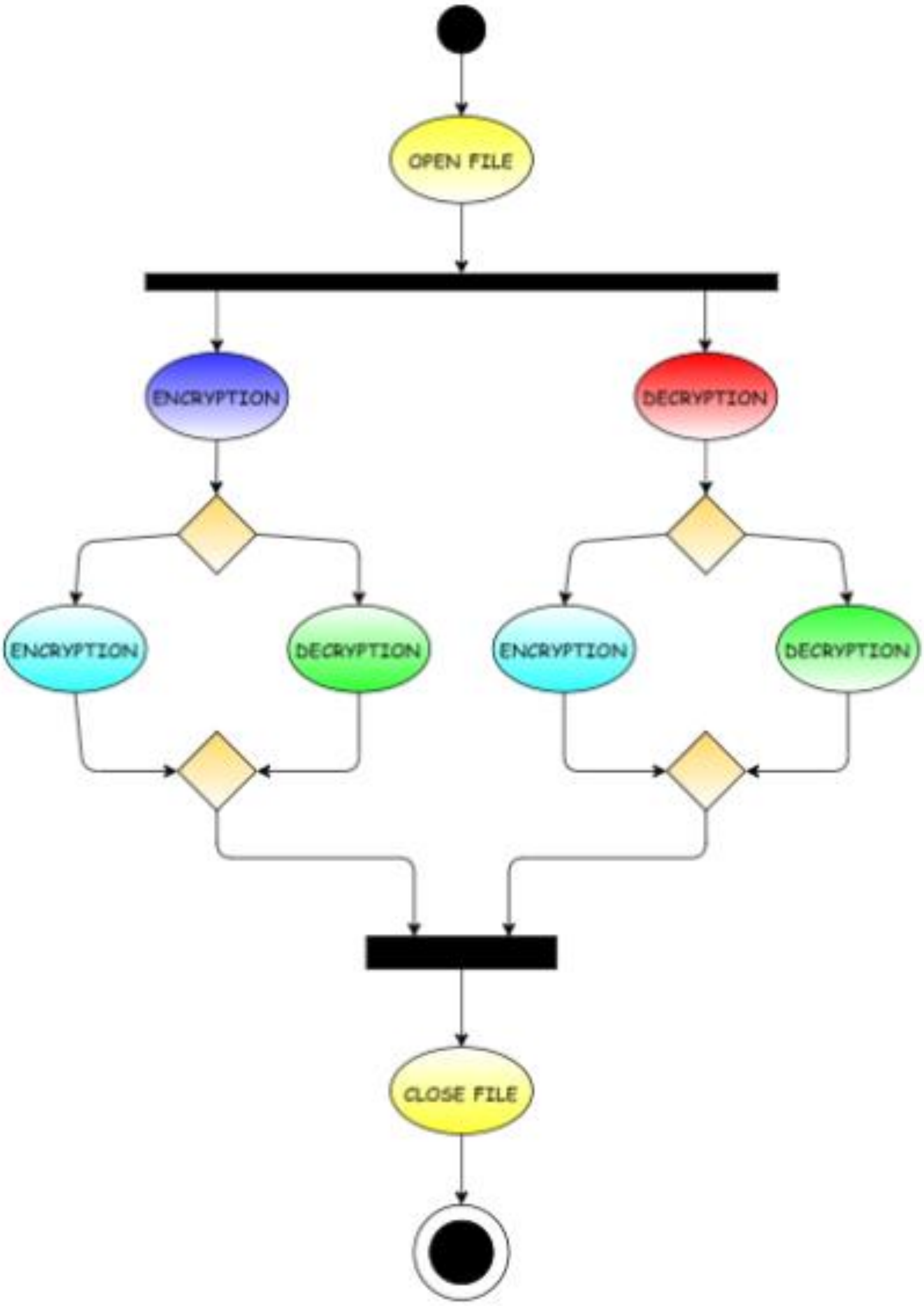




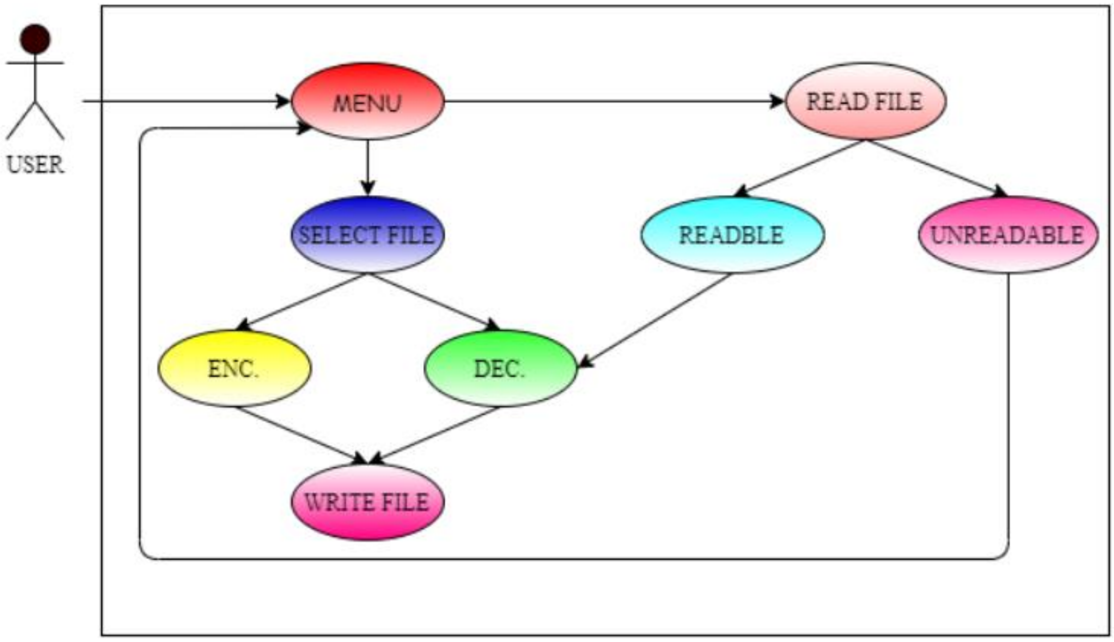
3.2.2 Architecture Design

The working purpose of the system is file encryption. In doing so, we plan to encrypt 128, 256, 512, 1024 or 2048 bit files using the AES encryption algorithm CTR mode, which is known as the most used symmetric and block encryption. By using CUDA, which is available as an add-on on the C programming language provided by NVIDIA for GPU / CPU, we increase the speed and decrease the cost of file encryption. If the files are small, we will use the CPU. If the files are large, we will use the GPU. With the created menu, the file selected by the user can be encrypted using AES encryption CTR mode. If the selected file is encrypted, the file is also decrypted and made readable.

3.2.3 Activity Diagram



3.3 Use Case Realization



4.Test Plan

4.1 Introduction

4.1.1 Version Control

Version No	Description of Changes	Date
1.0	First Version	May 29, 2020

4.1.2 Overview

In our project, we conducted a time-performance test with various file sizes. By applying encryption to these files, we tested their performance on CPU and GPU. We have determined which hardware is more effective in which file size.

4.1.3 Scope

This document includes the test plan of the use cases, test design specifications and the test cases correspond to test plan.

4.1.4 Terminology

Acronym	Definition
CTR_CPU	AES – CTR Mode Encryption with CPU
CTR_GPU	AES – CTR Mode Encryption with GPU

4.2 Features to be Tested

This section lists and gives a brief description of all the major features to be tested. For each major feature there will be a Test Design Specification added at the end of this document.

4.2.1 AES – CTR Mode Encryption with CPU(CTR_CPU)

In the first step of the test part of our project, we performed the encryption of files of different sizes on the CPU using Counter Mode (CTR) together with the AES-256 algorithm, which is one of the encryption methods.

4.2.2 AES – CTR Mode Encryption with GPU(CTR_GPU)

In the second step of the test part of our project, we performed the encryption of files of different sizes on the GPU using Counter Mode (CTR) together with the AES-256 algorithm, which is one of the encryption methods.

4.3 References

[1] SRS, 27 December 2019

[2] SDD, 16 February 2020

4.4 Test Design Specifications

4.4.1 AES – CTR MODE Encryption with CPU(CTR_CPU)

4.4.1.1 Subfeatures to be tested

4.4.1.1.1 1-KB Text File in CPU(CTR_CPU.1_KB)

The encryption time of the 1-KB file is 0,58 miliseconds in the CPU.

4.4.1.1.2 2-KB Text File in CPU(CTR_CPU.2_KB)

The encryption time of the 2-KB file is 0,627 miliseconds in the CPU.

4.4.1.1.3 4-KB Text File in CPU(CTR_CPU.4_KB)

The encryption time of the 4-KB file is 1,885 miliseconds in the CPU.

4.4.1.1.4 8-KB Text File in CPU(CTR_CPU.8_KB)

The encryption time of the 8-KB file is 3,421 miliseconds in the CPU.

4.4.1.1.5 16-KB Text File in CPU(CTR_CPU.16_KB)

The encryption time of the 16-KB file is 5,010 miliseconds in the CPU.

4.4.1.1.6 32-KB Text File in CPU(CTR_CPU.32_KB)

The encryption time of the 32-KB file is 6,665 miliseconds in the CPU.

4.4.1.1.7 64-KB Text File in CPU(CTR_CPU.64_KB)

The encryption time of the 64-KB file is 14,96 miliseconds in the CPU.

4.4.1.1.8 128-KB Text File in CPU(CTR_CPU.128_KB)

The encryption time of the 128-KB file is 31,784 miliseconds in the CPU.

4.4.1.1.9 256-KB Text File in CPU(CTR_CPU.256_KB)

The encryption time of the 256-KB file is 78,113 miliseconds in the CPU.

4.4.1.1.10 512-KB Text File in CPU(CTR_CPU.512_KB)

The encryption time of the 512-KB file is 112,814 miliseconds in the CPU.

4.4.1.1.11 1-MB Text File in CPU(CTR_CPU.1_MB)

The encryption time of the 1-MB file is 229,696 miliseconds in the CPU.

4.4.1.1.12 2-MB Text File in CPU(CTR_CPU.2_MB)

The encryption time of the 2-MB file is 454,108 miliseconds in the CPU.

4.4.1.1.13 4-MB Text File in CPU(CTR_CPU.4_MB)

The encryption time of the 4-MB file is 885,344 miliseconds in the CPU.

4.4.2 AES – CTR MODE Encryption with GPU(CTR_GPU)

4.4.2.1 Subfeatures to be tested

4.4.2.1.1 1-KB Text File in GPU(CTR_GPU.1_KB)

The encryption time of the 1-KB file is 57,782 miliseconds in the GPU.

4.4.2.1.2 2-KB Text File in GPU(CTR_GPU.2_KB)

The encryption time of the 2-KB file is 61,198 miliseconds in the GPU.

4.4.2.1.3 4-KB Text File in GPU(CTR_GPU.4_KB)

The encryption time of the 4-KB file is 65,386 miliseconds in the GPU.

4.4.2.1.4 8-KB Text File in GPU(CTR_GPU.8_KB)

The encryption time of the 8-KB file is 73,013 miliseconds in the GPU.

4.4.2.1.5 16-KB Text File in GPU(CTR_GPU.16_KB)

The encryption time of the 16-KB file is 73,915 miliseconds in the GPU.

4.4.2.1.6 32-KB Text File in GPU(CTR_GPU.32_KB)

The encryption time of the 32-KB file is 75,20 miliseconds in the GPU.

4.4.2.1.7 64-KB Text File in GPU(CTR_GPU.64_KB)

The encryption time of the 64-KB file is 76,002 miliseconds in the GPU.

4.4.2.1.8 128-KB Text File in GPU(CTR_GPU.128_KB)

The encryption time of the 128-KB file is 79,03 miliseconds in the GPU.

4.4.2.1.9 256-KB Text File in GPU(CTR_GPU.256_KB)

The encryption time of the 256-KB file is 80,80 miliseconds in the GPU.

4.4.2.1.10 512-KB Text File in GPU(CTR_GPU.512_KB)

The encryption time of the 512-KB file is 81,53 miliseconds in the GPU.

4.4.2.1.11 1-MB Text File in GPU(CTR_GPU.1_MB)

The encryption time of the 1-MB file is 83,113 miliseconds in the GPU.

4.4.2.1.12 2-MB Text File in GPU(CTR_GPU.2_MB)

The encryption time of the 2-MB file is 86,989 miliseconds in the GPU.

4.4.2.1.13 4-MB Text File in GPU(CTR_GPU.4_MB)

The encryption time of the 4-MB file is 88,07 miliseconds in the GPU.

5. References

[1] https://nezihesozen.github.io/mydoc/cuda2

[2] https://www.binance.vision/security/what-is-symmetric-key-cryptography

[3] https://www.garykessler.net/library/crypto.html

[4] https://www.tutorialspoint.com/cryptography/advanced_encryption_standard.htm

[5] https://www.tutorialspoint.com/cryptography/advanced_encryption_standard.htm

[6] https://www.highgo.ca/2019/08/08/the-difference-in-five-modes-in-the-aes-encryption-algorithm/

[7] https://www.kaspersky.com/resource-center/definitions/brute-force-attack

[8] https://www.oksbdc.org/why-is-technology-important-in-business/

[9] https://mainstreetpractitioner.org/tech-topics/keeping-your-data-safe/

[10] https://www.tutorialspoint.com/cryptography/advanced_encryption_standard.htm

[11] https://www.highgo.ca/2019/08/08/the-difference-in-five-modes-in-the-aes-encryption-algorithm/