1. Introduction

In this introductory chapter we will first present the motivation for this thesis and previous works done on this field and state what do we aim to achieve with this thesis. Finally we will also provide information regarding further organization of this thesis.

# Motivation

Before personal computers were available to the general public and certain industry, all the communications and interactions are done using a physical manner. All the technical systems are usually serviced and maintained physically commonly using human sense thus allowing it to be comprehended by broad public. Being up close with the systems allows a simple system of trust to be developed. Back then the main communications medium used plain copper cables which is managed by switch board operators that manually connect users through the switchboard and are trusted on by the assumption that the content of the conversation are secured as these services are provided by well known companies. Documents and contracts are produced using type writers and are signed and certified in the presence of a witness to authenticate it. However during the past 50 years numerous technological advancements has allowed computers to available to the masses and the availability of the internet has provided a reliable and fast communications platform either for personal or commercial uses. But with these advancements there is a price, the interconnectivity of the internet that allows us to communicate with people all over the world has also allowed people all over the world to eavesdrop to our communication if they have to means to do it. The similar level of trust exhibited with the physical communication is not adopted properly in regards of digital communication. Physical interactions or transactions are getting replaced online services such as face to face banking is replaced with online banking, and with the booming sector of online shopping a need to implement measures that provide a two end point communication. Most systems nowadays use both public and private key cryptography to counter the threats.

Multiple cryptography algorithms are introduced as a solution for the threats. Cryptography is a combination of arts and science that protect the message or known as plaintext from unintended recipients by encrypting/encoding the message using algorithms to convert the message into a ciphertext that can only be decrypted using a cipher or more commonly known as a key. The earliest use of cryptography can be traced to 400 BC used by the Spartans to allow military commanders to convey secret messages or battle instructions to each other this is done by writing on a piece of parchment that is wrapped on a baton while unwrapped the characters appear to be jumbled up and makes no sense. In the modern era, cryptography[AA05] is mostly used to encrypt digital data using encryption algorithms. Multiple encryption methods has been implemented for example Asymmetric encryption, Symmetric encryption, Hash Functions.

Computations in the old days are generally done by a Central Processing Unit(CPU) with a single core, due to the increase in demand for processing power intel and amd has developed multicore-CPU allowing multiple cores to be fitted onto 1 CPU. With the introduction of multicore-CPU the performance has improved by leaps and bounds allowing parallel programming and multi tasking capabilities. But there is still a limit to the number of cores u can place within a CPU. The performance of graphics cards are continuously improved as demand for high end graphics card specifically for gaming increases which surpasses Moore’s law. The untapped potential of the graphics cards are gradually discovered and in order to satisfy the markets demand for computing power in the year 2007 NVIDIA Corporation a graphics card manufacturing company released a new system called CUDA (Compute Unified Device Architecture) which allows the user to perform other computations other than graphical processing on the Graphics Processing Unit(GPU). Unlike CPU the GPU has a parallel throughput architecture that focuses on executing many concurrent threads at a slower speed compared to executing a single thread at higher speed as GPU is used for graphics rendering. The use of the GPU as a cryptographic accelerators to substitute dedicated hardware is an economical solution as the GPU does not cost as much.

The need for efficient cryptographic accelerators and the developments of CUDA in the recent years is a source of motivation for this thesis. The graphics card hold a lot of computing power when utilised correctly will produce great results.

# Previous Work And Aim Of This Thesis

The most common cryptographic protocol used is the Rivest Shamir Adleman (RSA) algorithm which is used to generate digital signatures. However the Elliptic Curve Cryptography(ECC) algorithm was introduced later then. The former was accepted as the standard cryptographic protocol for commercial use as it was introduced earlier and the patents for RSA has expired in the year 2000 hence it is widely adopted in a lot of cryptosystems. RSA computation involves exponentiations in the set of integers modulo some modulus as most expensive step. The latter one is based on the scalar multiplication of points on elliptic curves. Both of them requires a lot of computation power. In order to improve the performance numerous researchers has started to explore alternate techniques to decrease the time taken to complete the computation namely using dedicated hardware or by using the GPU. Extensive work has been done on the implementation of RSA algorithm on GPU by groups such as Moss et al. and Fleissner[MPS07][Fle07]. Some work has been done by various authors to implement Elliptic Curve Cryptography by using the GPU as a hardware accelerator namely Eric M.Mahe, Aaron E.Cohen and Keshab K.Parthi [MC14][CP10]. This thesis will focus on the research of the ECC algorithm to find out its properties and perform a performance comparison between implementations of ECC in serial and in CUDA. To the best of our knowledge there is no publication about a performance comparison for ECC in serial and CUDA. The aim of this thesis will focus on the research of the ECC algorithm to find out its properties, strengths, weaknesses and perform a performance comparison between implementations of ECC in serial and in CUDA. Besides that we also aim to create a foundation for the use of GPU as accelerators for public key cryptography through this thesis.

# 1.3 Thesis Outline

Chapter 2 will introduce mathematics used throughout this thesis and clarify our notation. Second, it will provide a rough overview of the cryptographic primitives that will gain more efficiency from our work. Besides that we will also be giving a short introduction to the GPU’design and its programming model. A brief explanation of the GPU’s hardware’s architecture will be given as well as the toolchain of Nvidia’s Compute Unified Device Architecture(CUDA). In the following Chapter 3 we will be writing on our proposed work and how do we plan to obtain the benchmark results. Chapter 4 will clarify the layers needed and analyse current techniques for their suitability to the GPU platform. Out of this first phase, we selected a couple of algorithms that sound promising from a theoretical standpoint. Basically we can differentiate three methods to do efficient modular arithmetic: multiplication using Montgomery’s technique, arithmetic in residue number systems (RNS) and special reduction techniques for moduli known as generalised Mersenne primes. Chapter 5 will give details on the implementation of the candidate algorithms from phase one. Then, we have benchmarked the result of this second phase and chose one principal candidate per group for further low-level optimisations. Finally, we conclude our thesis by comparing the throughput and latency results of our work to previous implementations in hard- and software.

2.0 Preliminaries

In this following chapter we will be going through some of the fundamental mathematics that is involved in the cryptographic algorithms and the notations that will be used throughout this thesis as well. We will first be going though modular arithmetic, cyclic groups and finite fields. Several prominent cryptographic systems will be introduced namely RSA, and ECC. Last but not least we will be going through an in-depth review on how Elliptic Curve is used as a cryptographic algorithm and its properties as the main focus of this thesis.

# 2.1 Modulo Arithmetic

[MVV96] Let N be a positive integer. If a and b are integers, then a is said to be congruent to b modulo N, written , if N divides (a - b). The integer N is called the modulus of the congruence. i.e.. since .For the sake of readability we will sometimes use a second, shorter notation to write a mod N in following chapters:

The equivalence class of an integer a is the set of all integers congruent to a modulo N. Now, if a = , where, then a = r (mod N). Hence each integer a is congruent modulo N to a unique integer between 0 and N- 1, called the least residue of a modulo N. Thus a and r are in the same equivalence class, and so r may simply be used to represent this equivalence class. The integers modulo N, denoted   , is the set of (equivalence classes of) integers {0, 1, 2, . . . , N - 1}. Addition, subtraction and multiplication in are performed modulo N.

# 2.2 Cyclic Groups

[MVV96] A group (\mathbb{G}, \*) consists of a set \mathbb{G} with a binary operation \* on \mathbb{G} satisfying the following properties:

(i) The group operation is associative. That is, for all a, b, c ∈ \mathbb{G}.

(ii) There is an element 1 ∈ \mathbb{G}, called the identity element, such that for all a ∈ G.

(iii) For each a ∈ \mathbb{G} there exists an element ∈ \mathbb{G}, called the inverse of a, such that .

A group \mathbb{G} is abelian or also known as commutative if, furthermore,

(iv) for all a, b ∈ \mathbb{G}.

A Group \mathbb{G} is finite if |\mathbb{G}| is finite. The number or elements in a finite group is called its order.

A Group \mathbb{G} is cyclic if there is an element such that for each there

is an integer i with . Such an element a is called a generator of \mathbb{G}. If \mathbb{G} is a group and , then the set of all powers of forms a cyclic subgroup of \mathbb{G}, called the subgroup generated by , and denoted by .

# 2.3 Finite Fields

[MVV96][HMV03][PP09] A Finite field also known as a Galois field is a field \mathbb{F} which contains a finite number of elements for which we can perform operations such as addition, multiplication, subtraction and inversion. A field \mathbb{F} is a set of elements that possesses these properties which is all elements within the field \mathbb{F} form an additive group with the group operation “” and the neutral element , all elements of \mathbb{F} except 0 form a multiplicative group with the group operation “” and the neutral element 1 and finally when two group operations are performed the distributive law holds for which,: The order of \mathbb{F} is the number of elements within the field \mathbb{F}. if \mathbb{F} is a finite field then **\mathbb{F}** contains elements for some prime and integer For every prime power order , there is a unique finite field of order . This field is denoted by, or sometimes by GF ().

## 2.3.1 Field operations

A field \mathbb{F} is equipped with two functions which is addition and multiplication. Subtraction of field elements is defined in terms of addition : for , where is a unique element in field \mathbb{F} which is also the negative element of . Division of field elements is defined in terms of multiplication : for , where is a unique element in field \mathbb{F}which is also the inverse element of

## 2.3.2 Prime Fields

A field \mathbb{F} is called a prime field when it satisfy certain characteristics. There exists a finite field \mathbb{F} of order if and only if is a prime power, i.e., where is a prime number called the characteristic of \mathbb{F}, and is a positive integer. If , then \mathbb{F} is called a prime field or commonly denoted as . Operations in a Prime Field are performed using modulo . For any integer , shall denote the unique integer remainder for which , which is obtained by dividing with also known as reduction modulo .

## 2.3.3 Binary Fields

Finite fields of order are called binary fields or characteristic-two finite fields. One way to construct is to use a polynomial basis representation. Here, the elements of are the binary polynomials (polynomials whose coefficients are in the field = { 0,1}) of degree at most m - 1:

An irreducible binary polynomial of degree Is chosen Irreducibility of means that cannot be factored as a product of binary polynomials each of degree less than m. Addition of field elements is the usual addition of polynomials, with coefficient arithmetic performed modulo 2. Multiplication of field elements is performed modulo the reduction polynomial For any binary polynomial , mod shall denote the unique remainder polynomial of degree less than m obtained upon long division of by this operation is called reduction modulo .

# 2.4 Elliptic Curve

An Elliptic Curve *E* is defined as a curve over a field \mathbb{F}. To allow easier understanding we will curves over real numbers \mathbb{R}.

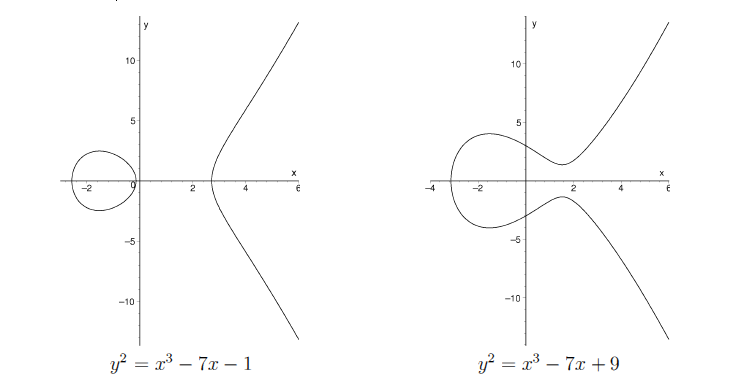


Figure 2.1 Elliptic Curve over \mathbb{R}.

An elliptic curve over a field \mathbb{F}, with characteristics not equal to 2 or 3 is defined by an equation in of the simplified Weierstraß form:

Where and satisfy A pair where , is a point on the curve if satisfies equation 2.1. The point at infinity, denoted by , is also said to be on the curve [HMV03]. The set of all points on E is denoted by E(F). The number of points in E(\mathbb{F}), denoted #E(\mathbb{F}), is called the order of E over \mathbb{F} or the group cardinality. Two example curves can be seen in Figure 2.1.

There is a chord-and-tangent rule for adding two points in E(\mathbb{F})resulting in a third point in E(\mathbb{F}). Together with this addition operation, the set of points E(\mathbb{F}) forms an abelian group with serving as its identity. Figure 2.2 depicts an exemplary point doubling and addition.

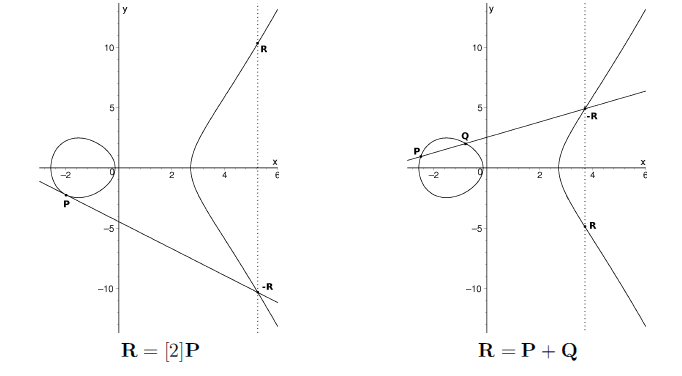


Figure 2.2 Point Doubling and Addition on Elliptic Curve over \mathbb{R}

Let **P** = and **Q** = be two distinct points on an elliptic curve E. Then the sum **R** of **P** and **Q** is defined as follows: First draw a line through **P** and **Q**; this line intersects the elliptic curve at a third point. Then **R** is the reflection of this point about the x-axis. The double **R** of **P** is defined as follows. First draw the tangent line to the elliptic curve at **P**. This line intersects the curve at a second point. Then **R** is the reflection of this point about the x-axis.

## 2.4.1 Affine Coordinates

Based on the geometric representation above algebraic formulas for \mathbb{F} where and is devised to calculate the points.[MIS10]

First a short introduction to a negative point. If then where is known as the point at infinity. The point (x, -y) is denoted by and is called the negative of . Note that is indeed a point in as well as.

Take a look at **Point Addition**. Let and , where . Then , where

Thus, a point addition or doubling costs 1 inversion, 2 multiplications and 1 squaring, and 1 inversion, 2 multiplications and 2 squarings, respectively. While the formulas for affine coordinates presented above could be easily implemented, they might impose a serious performance drawback if the inversions needed to compute λ are significantly more expensive than multiplications in the underlying field \mathbb{F}. In this case it might be advantageous to use different coordinate systems or a completely different description of the curve.

## 2.4.2 Jacobian Projective Coordinates

We can use a special case of projective coordinates known as the Jacobian Projective Coordinates that makes use of a three dimensional coordinate system to avoid field inversions hence reducing the performance drawback. The formulae for projective coordinates can be derived from the (affine) Weierstraß form by substitution. A Jacobian Projective Coordinate the curve conventional Weierstraß is converted to

The projective point on E corresponds to the affine point when and to the point at infinity (1 : 1 : 0) otherwise. The negative of is [CF05].

Let us reexamine **Point Addition**. Let and , where Then where

Thus, a point addition costs 12 multiplications and 4 squarings in \mathbb{F}. If one of the points, say Q, is given in affine coordinates, i.e. = 1, this can be reduced to 8 multiplications and 3 squarings. This technique is known as using mixed affine-Jacobian coordinates.

# 2.5 Cryptosystems

In this section we will introduce some prominent cryptosystems that are currently in use today. The cryptosystem that we will be introducing make use of asymmetric cryptography which is also known as public key cryptography. A brief introduction to asymmetric cryptography, it is a type of cryptography that makes use of 2 distinct keys for which one is a public key that is shared while the other is a private key which is kept secret.

Firstly we will be introducing the Diffie-Hellman cryptosystem which is considered as the pioneer for public key cryptography. Next we will then introduce the Rivest-Shamir-Adleman (RSA) cryptosystem that is used for encryption and digital signature generation. Finally we will be introducing Elliptic Curve Cryptosystem (ECC) which is the latest addition to the cryptosystem family. Due to its versatile nature is can be used to perform encryption, digital signature generation and key exchange.

# 2.5.1 Diffie-Hellman cryptosystem

This is the first public-key cryptosystem that was invented by Whitfield Diffie and Martin Hellman, working in collaboration with Ralph Merkle in 1976[CWQ07]. This cryptosystem uses two different keys a public key and a secret key. Both keys are secretly generated. The use of both keys allowed numerous applications to be made possible such as serving as Secure Shell (SSL), Transport Layer Security (TLS) and Internet Protocol security (IPsec). The Diffie-Hellman algorithm is based on the difficulty of the discrete logarithm problem where the main idea is the exponentiation in [PP09][Kam10].

The value k is the joined secret created to perform operations such as encryption or used as a session key. In order to perform any Diffie-Hellman operations we must first obtain the domain parameters which is a common integer that both parties must agree to use and where is a large prime number and . A fictional character Alice will then select a secret key and produce the public key by choosing another random integer where and calculating . Another fictional character Bob will do the same which a secret key hence producing a public key . After both parties exchange their public keys they are able to obtain the same session key and which results in the same .

Even though Diffie achieved the concept of an asymmetric cipher, Diffie-Hellman cryptosystems are still susceptible to attacks such as Man in the middle attacks where the attacker intercepts both key exchange messages and is able to reverse engineer and obtain both secret keys of Alice and Bob. However, he inspired other mathematicians and scientists to discover another cipher, the RSA cryptosystem which is discussed next.

# 2.5.2 Rivest-Shamir-Adleman cryptosystem

The Rivest-Shamir-Adleman (RSA) cryptosystem is the first publically practicable cryptosystem introduced in the year 1977. RSA uses two keys one is a public key while the other is a private key. The user of RSA will first create a public key based by multiplying two distinct prime numbers for which . After that by using the Euler’s totient function is calculated using After is calculated an integer which is also known as the public key exponent is chosen for: gcd. A value also known as multiplicative inverse of is calculated using. To summarise all the public key for RSA is and the secret key which we must keep as a secret will be ) but values will also be kept secret as they can be used to calculate the value .

RSA Key Generation Algorithm

Require: The security parameter , i.e. the length of in bits

Ensure: RSA key pair(

1. where
2. where
3. Return (

After all the values are obtained the encryption is done by converting the message M into an integer . By applying this formula we are able to obtain the ciphertext . By applying the decryption formula the original message can be obtained. Currently the applications for RSA are mostly used when the is a need for low cost key generation such as Smart Cards where the low cost key generation will produce higher efficiency and acceptable level of security[Ven04]. However RSA is still vulnerable to certain types of attacks such as mathematical attacks or brute force attacks that target specifically the decryption key which is by performing factorisation on and easily reconstruct [Bon99]. Besides that RSA is also vulnerable side channel attacks which targets the physical channel such as power consumptions or timing behaviours because by observing the behaviour of the we can deduce private key from the operations that are currently performing as the main computational load of RSA is usually done for the squarings and multiplications hence a single squaring operation will result in a value 0 and a combination of squaring and multiplication will result in a value 1[PP09].



Figure 2.3 Encryption and Decryption for RSA algorithm

# 2.5.3 Elliptic Curve Cryptography

The initial discovery of Elliptic Curve Cryptography (ECC) was made by Neal Koblitz and Victor S.Miller independently in the year 1985 and 1987 [Mil85][Kob87]. It is proposed as an alternative cryptography system as appose to RSA. Instead of performing computations in a multiplicative group we can utilize E(\mathbb{F}) where the parameters are smaller hence providing a better performance. Besides that the level of security that ECC provides is on par with other cryptosystems while using a significantly shorter key length[LD00].

|  |  |  |  |
| --- | --- | --- | --- |
| Symetric Key Length | Example Algorithm | ECC Key Length For Equivalent Security | RSA Key Length For Equivalent Security |
| 80 | SKIPJACK | 160 | 1024 |
| 112 | Triple-DES | 224 | 2048 |
| 128 | 128-bit AES | 256 | 3072 |
| 192 | 192-bit AES | 384 | 7680 |
| 256 | 256-bit AES | 512 | 15360 |

Table 1: ECC,RSA key length comparison

Based on the introduction we had above on Elliptic Curve we can utilize its properties to form an Elliptic Curve discrete logarithm problem (ECDLP) where it is another variation of the discrete logarithm problem as EDCLP is defined over a point of an elliptic curve. The security aspect of ECC depends on a EDCLP where the discrete logarithm problem is applied on a elliptic curve [AK04].

In order to setup the discrete logarithm problem for ECC we must first determine the order of the group which is the number of elements that resides on the curve. The task of finding out the exact number of elements on the curve is an elaborate task but we can use *Hasse’s theorem* to obtain a rough estimation [Tol09]. *Hasse’s theorem* states that if E is an elliptic curve over a prime field , let #E(\mathbb{F}) be the number of points on the curve E then:

and

After determining the number of elements that reside of the curve we can now begin to setup the ECDLP. The definition for ECDLP is given E is an elliptic curve over a prime field , there is a primitive element and another element where:

Hence the discrete logarithm problem is to find where . By obtaining the value which is an integer it can be used as a private key while the point with coordinate , ) is used as the public key. In contrast to cases where the discrete logarithm problem is done on where both keys are integers, the operations describe above is known as point multiplication which is different from conventional multiplication where you can directly multiply the integer with the point on the curve. The operations used to calculate the points are similar to the ones describe in section 2.4 where we make use of point addition and point doubling to determine the coordinate of point .

An example where we will use the double-and-add algorithm find the point which has a notation of where is the generator element which has the coordinate and satisfy this equation . We will first give a brief introduction to the algorithm. The algorithm makes use of the binary representation of which is also the private key in binary form and the generating element in binary form. We will scan the bit representation of the scalar value from left to right and perform doubling operations during every iteration and only perform an additional addition operation when the current bit is of value . A full workout is shown below where we are able to obtain the coordinates of point t.

The algorithm will first start off with the left most bit which is and ending with the right most bit which is .

|  |  |  |
| --- | --- | --- |
| Steps | Operations | Description |
| 1 |  | This is the initial setting. Bit processed = |
| 2 |  | Point Doubling is executed. Bit processed = |
| 2.b |  | Since is of value 1 then Point Addition is executed. Bit processed = |
| 3 |  | Point Doubling is executed. Since bit is of value 0 we will then move on the next bit. Bit processed = . |
| 4 |  | Point Doubling is executed. Bit processed = . |
| 4.b |  | Since is of value 1 then Point Addition is executed. Bit processed = . |

Table 2 Steps to determine the types of operations used to obtain coordinate of point

From the table above we are able to obtain the coordinate of point t by performing DADDA operations in sequence for which D= doubling and A= addition.

## PROOF

,

when 6,

,

when 10,

,

when 16,

,

when ,

,

when 16,

Based on the calculations done above we can clearly see that all the points satisfy the equation . By using this property we effectively compute which is basically the number of hops that we make on the curve. In ECC we will publish the generating element which is essentially point and the end product of the number of hops point as the public parameters and public key and use the number of hops as the private key. In order to break the cryptosystem the attacker basically has to determine the number of hope that we made.

As explained in section 2.4.1 and 2.4.2 there are 2 types of coordinates that we can use to calculate the point addition and point doubling which is using the affine coordinate ( or the Jacobian projective coordinate . Due to the nature of modular arithmetic the most time consuming operation is modulo inverse operation for which the time required to perform a modulo inversion is equivalent to performing 40 multiplications because in order to perform a modulo inversion we have to make use of the Euclidean extended algorithm where the number of multiplications involved in that algorithm is very expensive. Therefore in order to improve the performance the use of Jacobian projective coordinate is advised. The table below will show the cost of each operation where I = Inversion, M = multiplication and S = Squaring.

|  |  |  |
| --- | --- | --- |
| Coordinate system | Point Doubling | Point Addition |
| Affine Coordinate | I+2M+1S | I+2M+2S |
| Jacobian Projective Coordinat | 4M+6S | 12M+4S |

Table 3 Field Operations Needed To Implement Elliptic Curve Operations[MAR08]

There are two types of finite field that an elliptic curve is often applied on which is prime fields and binary fields as explained in section 2.3. In order to ensure that the finite fields offer is up capable to provide a high level of security the National Institute of Standards and Technology has proposed the use of elliptic curves over prime fields, binary fields and Koblitz curves over a binary field. There is 5 recommended elliptic curves over prime fields, 5 recommended elliptic curves over binary fields and 5 recommended Koblitz curves over binary fields. Their sample parameters is shown in appendix section[NIST99][HMV03].

# 2.6 GPU Computing

In this section a short introduction to traditional Graphic Processing Unit (GPU) computing will be presented. After we will then take a look at the programming model for Nvidia’s Compute Unified Device Architecture (CUDA).

## 2.6.1 Traditional GPU Computing

Over the past 20 years the main focus of GPU manufacturers is to cater to the needs of the gaming community hence they dedicate a lot of their resources to produce GPU with an optimized GPU pipeline which consists of multiple stages which is transform and light, assemble primitives, rasterize and shader.



Figure 2.4 GPU pipeline abstraction.

As shown in figure 2.4 the pipeline begins when the application associated with CPU passes vertices which are the abstract 3D description of the scene that is to be rendered to the GPU. The vertices are then transformed into 2D and lighting is applied to the vertices. After that a screen-space description is assembled from the 2D vertices, the screen-space description it is made out of triangles as triangles always exists on a single plane. The screen-space description is then rasterized where the 2D space representation is converted into raster format and the pixel values are then produced. Finally the results are then fed to the shader which colours the pixels and determine which pixel can be seen from the current view also known as depth check. After going through the pipeline the results are then stored on the GPU buffer which can be displayed on the connected video output device[Ceb04] [VenND].

The first generation of GPU’s had all the functions needed to implement the pipeline hardwired but as time progresses the stages became more flexible which allows it to be programmable through the introduction of specialized processors such as vertex and fragment processors which greatly improves the flexibility of the transform and light stage and the shading stage.

As the performance increase while the price decreased the research community began to come out with ways to utilize the GPU to perform computation intensive tasks known after as General Purpose Computing on Graphic Processor(GPGPU). However due to the specialized nature of the processors and the application programming interface were built to implement the graphic pipeline the use of the GPU to perform computation intensive tasks generated a lot of overhead. These limitations has motivated GPU manufacturers to develop new APIs that is design specifically for the GPGPU community and introduce changes to their product in order to provide better support. ATI release a solution called Close To Metal(CTM) while Nvidia released CUDA.

## 2.6.2 CUDA Programming Model

## 2.6.2 CUDA Programming Model

CUDA is an abstract programming model. The general idea is that a computer known as a host will offload any computation intensive tasks to the GPU known as the device. The function that is to be executed in the device is known as a kernel. Both the host and the device have their own memory areas known as host memory and device memory. Part of the overhead incurred during the initial use of the GPU to perform computation intensive tasks is the transfer of data between the two memory spaces but this has been addressed when CUDA supplied optimized functions to transfer the data.

The table below will show one recommended curve for each field and their parameters[NIST99].

|  |
| --- |
| Curve P-192 over Prime Field  p = 62771017353866807638357894232076664160839087\ 00390324961279  r = 62771017353866807638357894231760590137671947\ 73182842284081  s = 3045ae6f c8422f64 ed579528 d38120ea e12196d5  c = 3099d2bb bfcb2538 542dcd5f b078b6ef 5f3d6fe2 c745de65  b = 64210519 e59c80e7 0fa7e9ab 72243049 feb8deec c146b9b1  = 188da80e b03090f6 7cbf20eb 43a18800 f4ff0afd 82ff1012  = 07192b95 ffc8da78 631011ed 6b24cdd5 73f977a1 1e794811  p=prime modulus  r=order  s=160bit input seed to the SHA-1 algorithm  c is the output of the SHA -1 algorithm  b=coefficient which satisfy(  base point x  = base point y |

Table 3 Curve P-192 over Prime Field

|  |
| --- |
| Curve B-163 over Binary Field  r = 5846006549323611672814742442876390689256843201587  Polynomial Basis:  = 2 0a601907 b8c953ca 1481eb10 512f7874 4a3205fd  = 3 f0eba162 86a2d57e a0991168 d4994637 e8343e36  = 0 d51fbc6c 71a0094f a2cdd545 b11c5c0c 797324f1  Normal Basis:  s = 85e25bfe 5c86226c db12016f 7553f9d0 e693a268  = 6 645f3cac f1638e13 9c6cd13e f61734fb c9e3d9fb  = 0 311103c1 7167564a ce77ccb0 9c681f88 6ba54ee8  = 3 33ac13c6 447f2e67 613bf700 9daf98c8 7bb50c7f  r= base point order  =coefficient for polynomial basis  s= l60-bit input seed to the SHA-l based algorithm  =the output of the SHA-l based algorithm  =base point x  =base point y |

Table 4 Curve B-163 over Binary Field

[AA05]Aziz, M. and Akbar, S. (2005). Introduction to Cryptography. *2005 International Conference on Microelectronics*. [online] Available at: <http://dx.doi.org/10.1109/icm.2005.1590056>

[Mil85] Miller, V. (1986). Use of Elliptic Curves in Cryptography. *Advances in Cryptology — CRYPTO ’85 Proceedings*, pp.417-426.

[Kob87] Koblitz, N. (1987). Elliptic curve cryptosystems. *Math. Comp.*, 48(177), pp.203-203.

[MVV96] Alfred J. Menezes, Scott A. Vanstone, and Paul C. van Oorschot.

Handbook of Applied Cryptography. CRC Press, Inc., Boca Raton,

FL, USA, 1996.

[HMV03] Darrel Hankerson, Alfred J. Menezes, and Scott Vanstone. Guide

to Elliptic Curve Cryptography. Springer-Verlag New York, Inc.,

Secaucus, NJ, USA, 2003.

## [PP09] Paar, C. and Pelzl, J. (2009). *Understanding cryptography*. Berlin: Springer.

[Bon99] Boneh, Dan. "Twenty years of attacks on the RSA cryptosystem." *Notices of the AMS* 46.2 (1999): 203-213.

[Ven04]Venkat Suryadevara Efficient on-board RSA key generation with smart

cards, School Electrical Engineering and Computer Sciences, Oregon

State University, Corvallis, 2004,

[Fle07] Sebastian Fleissner. GPU-accelerated Montgomery exponentiation.

In 7th International Conference on Computational Science (ICCS

2007) (LNCS 4487), pages 213–220, May 2007.

[MPS07] Andrew Moss, Dan Page, and Nigel Smart. Executing modular exponentiation

on a graphics accelerator. Cryptology ePrint Archive,

Report 2007/187, 2007. http://eprint.iacr.org/.

[MC14] Mahe, E., & Chauvet, J. M. (2014). Fast GPGPU-Based Elliptic Curve Scalar Multiplication. *IACR Cryptology ePrint Archive*, *2014*, 198.

[CP06] Cohen, A. and Parhi, K. (2006). A New Side Channel Resistant Scalar Point Multiplication Method for Binary Elliptic Curves. *2006 Fortieth Asilomar Conference on Signals, Systems and Computers*.

[CP10] Cohen, A. E., & Parhi, K. K. (2010, August). GPU accelerated elliptic curve cryptography in GF (2 m). In *Circuits and Systems (MWSCAS), 2010 53rd IEEE International Midwest Symposium on* (pp. 57-60). IEEE.

[LD00] Lopez, J., & Dahab, R. (2000). An overview of elliptic curve cryptography.

[CF05] Henri Cohen and Gerhard Frey, editors. Handbook of elliptic and hyperelliptic curve cryptography. Chapman & Hall/CRC Press, Boca Raton, FL, USA, 2005.

[CWZ07] Charles Edge, William Barker & Zack Smith A brief history of cryptography, Foundation of Mac OS X Security, October 23 2007.

[Kam10] Kamil Abdullah Comparison between the RSA cryptosystem and elliptic curve cryptography, 2010

[MIS10] McGrew, D., Igoe, K., & Salter, M. (2010). Fundamental Elliptic Curve Cryptography Algorithms. *draft-mcgrew-fundamental-ecc-02 (work in progress)*.

[AK04] Amol Dabholkal & Kin Choong Yow Efficient implementation of elliptic

curve cryptography and personal digital assistance (PDAs), School of

Computer Engineering, Nanyang Technological University, Singapore,

2004.

[Tol09] Tolkov, I. (2009). *Counting points on elliptic curves: Hasse’s theorem and recent developments*. [online] Available at: <https://www.math.washington.edu/~morrow/336_09/papers/Igor.pdf>

[NIST99]National Institute of Standards and Technology. (July 1999) RECOMMENDED ELLIPTIC CURVES FOR FEDERAL GOVERNMENT USE. [online] Available at: http://csrc.nist.gov/groups/ST/toolkit/documents/dss/NISTReCur.pdf

## 

[Mar08]Martin, L. (2008). *Introduction to Identity-Based Encryption*. Norwood: Artech House.

# [Ceb04] Cebenoyan, C. (2004). Graphics pipeline performance. *GPU Gems*, 473-486.

[VenND]Venkatasubramanian, S. (n.d.). *Understanding thegraphics pipeline*. [online] Available at: http://www.seas.upenn.edu/~cis565/LECTURES/Lecture2%20New.pdf