Elliptic Curve Cryptography with Triple DES Encryption

Final Proposal

Members: Nico Bellante, Lucas Dahl, Manish Gupta, Xiong-Yao Zha

Class: ECE 337

Section: Wednesday 11:30

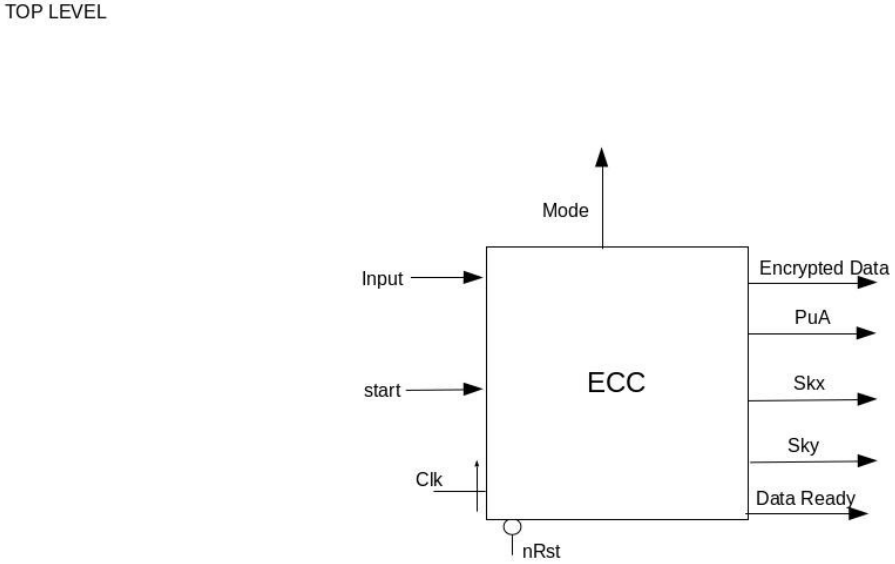
Due: March 23, 2015

Teaching Assistant: Yunus Akhtar

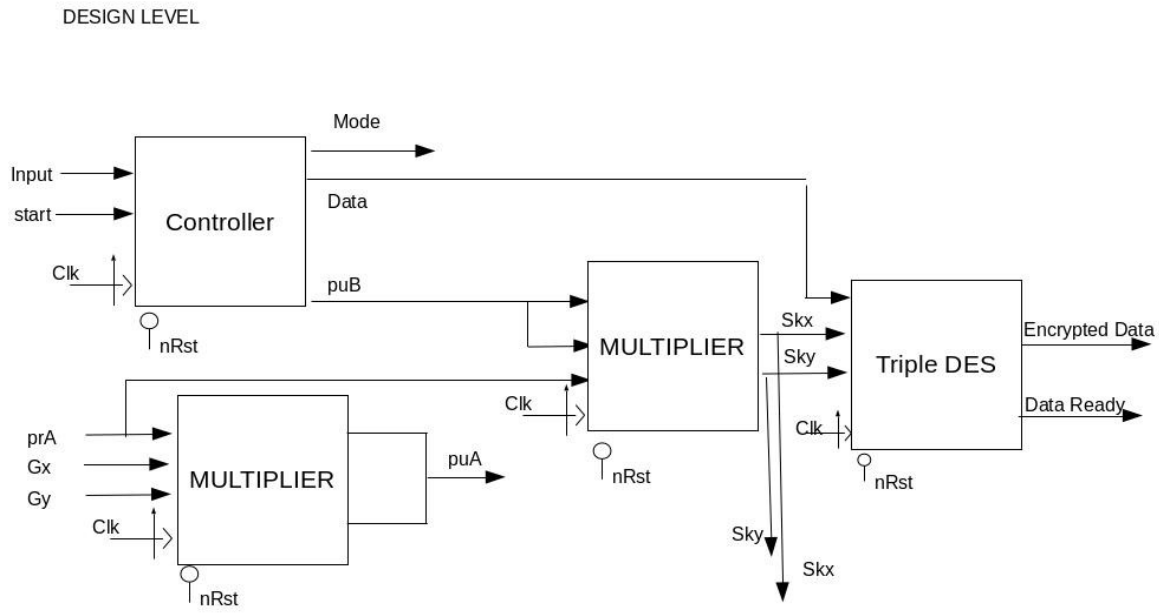
Executive Summary

The team will design and build a chip for elliptic curve cryptography public and private key generation. The chip will have a wide range of applications, and is especially useful for internet technicians and computer security enthusiasts. Some positive usages of the chip includes: generating the public key, and generating the private key. These functions are especially useful for low powered devices, as the key generation process for ECC is not as computationally intensive as other algorithms (ex. RSA). We are going to use the key generated for Triple DES Encryption. Another reason for its usefulness is that keys for ECC can be much smaller than the keys generated by other algorithms to achieve the same security. Even though the function of the chip can be accomplished in software, the ASIC version of it will allow for a much more efficient implementation of elliptic curve cryptography keys generation. The ASIC version is unique in that it utilizes hardware in a parallel manner to compute and verify points on the elliptic curve. The same can be said for triple DES encryption as most of the tasks can happen in parallel making it faster to run on hardware. The fundamental task that the chip will accomplish is to first receive the bitstream of the data over the wire. Upon receiving the data, it will generate a public and a private key according to the elliptic curve chosen. Generating keys for ECC require Galois Fields arithmetic. In the low level it involves binary addition, division, and multiplication, and in the high level it involves point addition and multiplication. The preliminary proposal will cover the high level block diagram of the chip, the inputs/outputs and functions of each sub-part of the chip, the architectural block diagram, and lastly the requirements and limitations that goes into building the elliptic curve cryptography key generation.

System Usage Diagram



Design Architecture



Elliptical Curve Cryptography Explanation

Elliptical Curve Cryptography is a more modern style of encryption that in essence is the addition of the same point many times. To add a point P by itself on a curve involves drawing a tangent, finding the second intersection point and its reflection is 2P. To add 2P and P, instead of drawing a tangent join the 2 points with a line and do the same. We take in the public key of B and use a constant random private Key for our chip A. The public Key B is a set of coordinates. For our purpose, we are mainly using ECC for key exchange and using the public key we get and our private key to create a session key that is hence used for Triple DES Encryption. We are also generating the public key for A for other chips to use.

Galois Field Explanation

Galois field are used to speed up the implementation of Elliptical curve cryptography in hardware. We are using GF(2^n) field with n being 163. This means that the highest bit position is going to be 163. These fields can be represented as polynomials from x^0 to x^163 with the coefficient being either 1 or 0. All calculations have to be done in a modular way. So for example if a number a is 000100, which is x^2 and the modular binary number a is 111, which is x^2 + x + 1, h = a%b can be got by setting x^2 + x + 1 = 0, and seeing that x^2 = (x+1) which
Different operations on these polynomials is explained later on.

Operational Characteristics and Functional Block Diagrams

Galois Field Multiplication

This block performs a Galois field multiplication. Multiplication in GF(2^n) works differently from normal binary multiplication. The multiplication involves multiple layers of shifting and xoring. For example if a is (00101), b is (01001) and m is (100100), and c=(a*b) mod m.

To multiply a by b, you iterate through each bit of b, if its a 1 shift i times where i is the index of the bit position and xor all values shifted in the end.
So since b[0] = 1, then shift a once
a1 = 01010

And also b[3] = 1, shift a by 1, then that a again by once and finally shift that again, i.e. Shift a once 3 times.
a2_1 = 01010
a2_2 = 10100
a2_3 = 101000

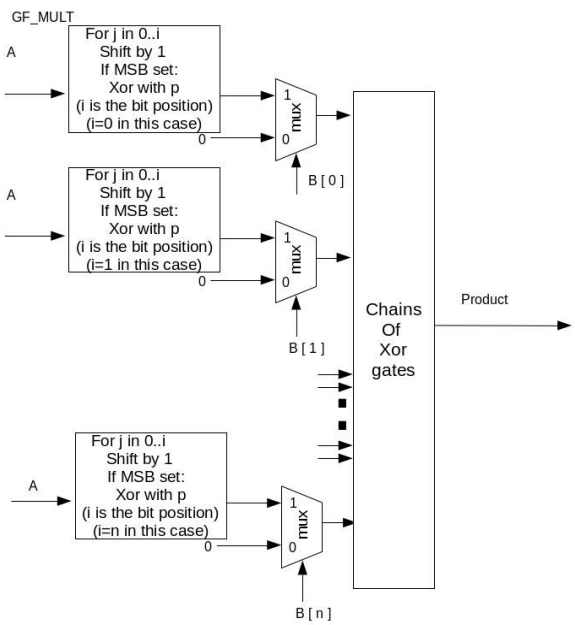
Since a2_3[5] == 1, and m[5] is the max power, we have to do a mod m. To do this we can use the fact that the additive inverse of 100100 is 100. This can be derived by doing the long polynomial division.

So a2 = 101000 xor 100
a2 = 101100

You just need to finally add a1 and a2 together to get the final c.
c = a1 xor a2
= 100110

In a general case you will have many layers of xor, we are going to do it in a logarithmic way so that it goes through faster.

The theory behind this has been tested using a python script.



Signal Name	Type	Number of bits	Description
A	IN	1	One element in the finite field of set G(2^n)
B	IN	1	One element in the finite field of set G(2^n)
Start	IN	1	Clear and start the multiplication
Product	OUT	1	The result of Galois field multiplication
Out	OUT	1	Indicates that the multiplication has completed, and the product is ready to be grabbed

Galois Field Division

This block performs a Galois field division.

a^((p^n)-1) = 1

a^(-1) = 1/a

= 1 * a^(-1)

= a^((p^n)-1) * a^(-1)

= a^((p^n)-2)

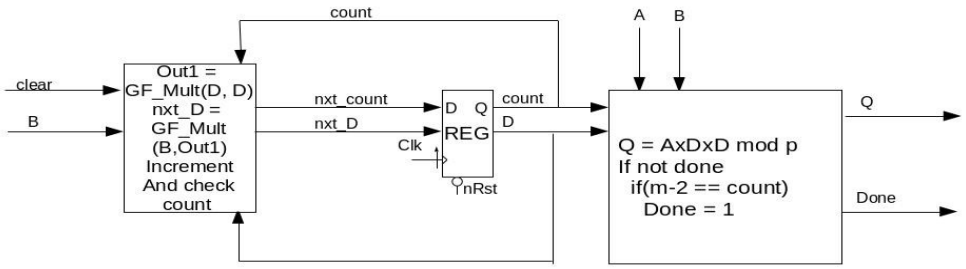
So to do D = A/B, we can do A*(B^-1)
To do the inverse, we do the following

D=B
for i = 0..m-2
D = B* (D * D)

//m-2 not inclusive, m = 163
B_MI = D * D
Q = A * B_MI
Q is the quotient

GF_DIVISION

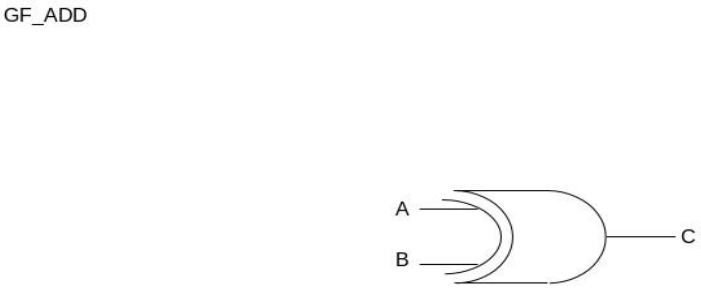
n=162 for our case



Signal Name	Type	Number of bits	Description
CLK	IN	1	The System Clock.
nRST	IN	1	This is an asynchronous, active-low system reset. When this line is low(logic ‘0’) , all registers/ flip-flops in the device must reset to their initial values
A	IN	1	Element of the Galois field.
B	IN	1	Element of the Galois field.
Clear	IN	1	Clear and start the division
Q	OUT	1	The quotient
Done	OUT	1	Indicates that the multiplication has completed, and the quotient is ready to be grabbed

Galois Field Addition

This block performs a Galois field addition. Quite simply an xor of the 2 inputs



Signal Name	Type	Number of bits	Description
A	IN	1	Element of the Galois field.
B	IN	1	Element of the Galois field.
C	OUT	1	Sum of the two elements

Point Addition

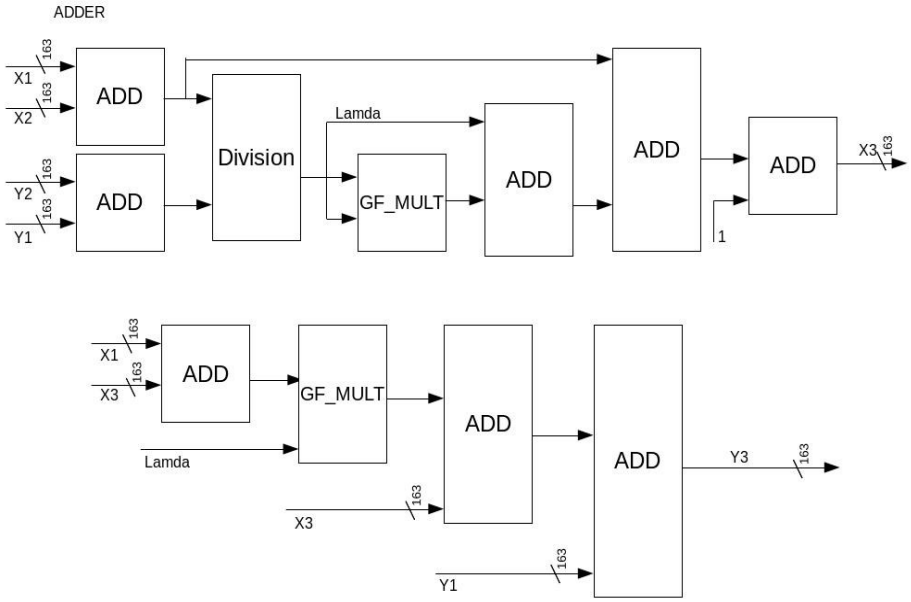
This block performs a point addition. Following the equations:

$m = (Y1 + Y2)/(X1 + X2)$

$X3 = m^2 + m + X1 + X2 + 1$

$Y3 = m(X1 + X3) + X3 + Y1$

With all operations referring to Galois field arithmetic



Signal Name	Type	Number of bits	Description
X1	IN	163	X coordinate of the first point
Y1	IN	163	Y coordinate of the first point
X2	IN	163	X coordinate of the second point
Y2	IN	163	Y coordinate of the second point
X3	OUT	163	X coordinate of the third point
Y3	OUT	163	Y coordinate of the third point

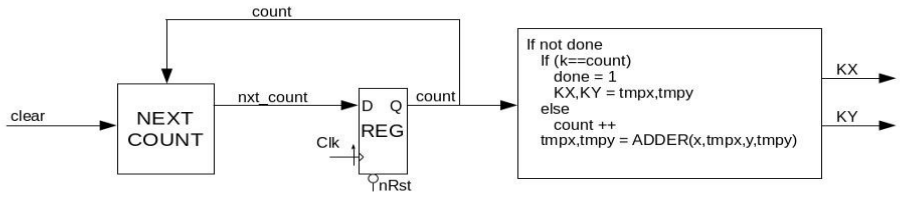
Point Multiplication

This block performs a point multiplication. Follow the equations:

k(X,Y) = (X,Y) + (X,Y) + ... + (X,Y) (k times)

With all additions referring to point addition:

MULTIPLIER



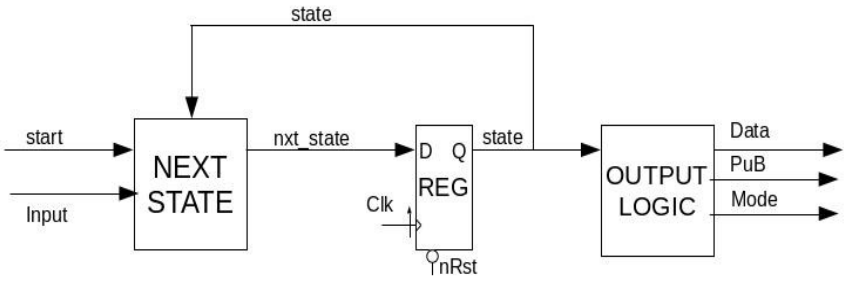
Signal Name	Type	Number of bits	Description
CLK	IN	1	The System Clock.
nRST	IN	1	This is an asynchronous, active-low system reset. When this line is low(logic ‘0’) , all registers/ flip-flops in the device must reset to their initial values
K	IN	1	Scalar value to multiply the first point by.
X1	IN	1	X coordinate of the first point

Y1	IN	1	Y coordinate of the first point
Clear	IN	1	Clear and start the multiplication
X2	OUT	1	X coordinate of the second point. Result of point multiplication.
Y2	OUT	1	Y coordinate of the second point. Result of point multiplication
Done	OUT	1	Indicates that the multiplication has completed, and the product is ready to be grabbed

Controller

Control the I/O operations for the chip. A state machine that handles the sequence of loading or outputting the generated keys.

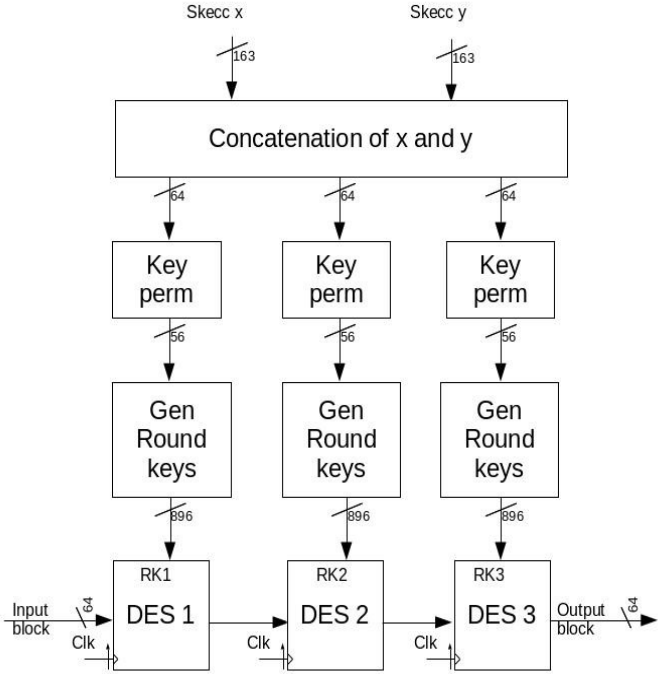
CONTROLLER



Signal Name	Type	Number of bits	Description
CLK	IN	1	The System Clock.
nRST	IN	1	This is an asynchronous, active-low system reset. When this line is low(logic ‘0’) all registers/ flip-flops in the device must reset to their initial values

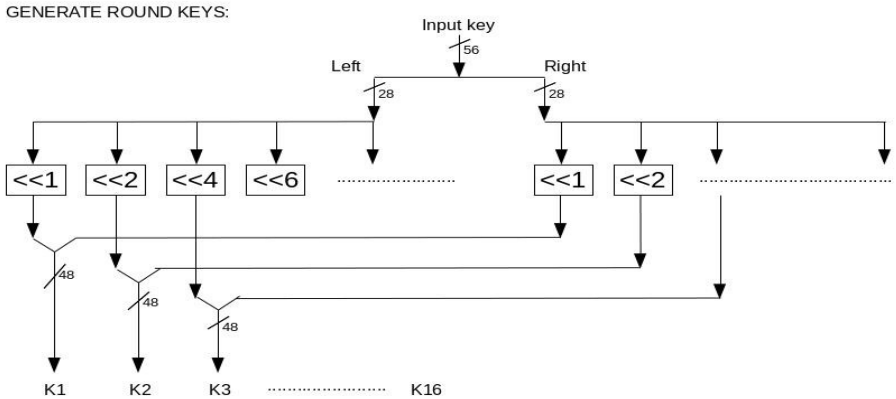
Start	IN	1	An active-high signal that indicates the public key b is ready, and also to indicate the start of key generation algorithm.
Key_ready	OUT	1	An active-high signal that indicates the completion of key generation, and to start outputting the generated keys.

TRIPLE DES



Generate Round Keys

Create round key from the input key by circular shifting and concatenating the left and right halves by multiples of 2.

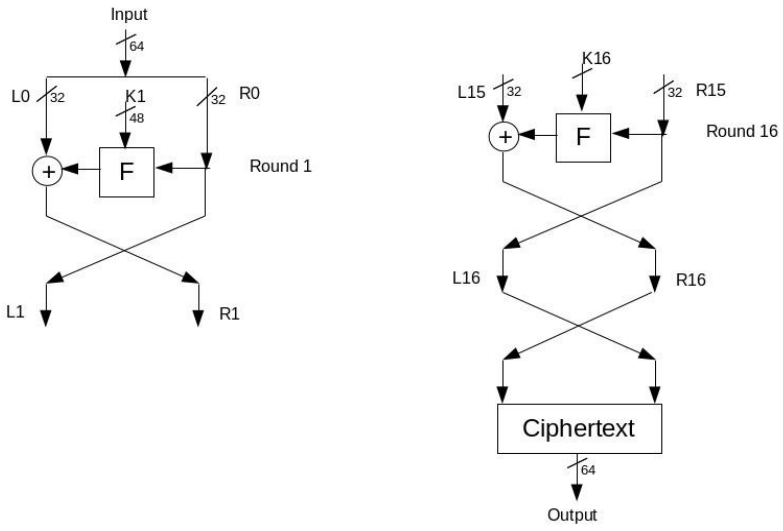


Signal Name	Type	Number of bits	Description
Input key	IN	56	The permuted key that is used to generate the round keys
K1..K16	IN	56	The generated round keys.

DES

Encrypt the data based on the round keys.

DES:

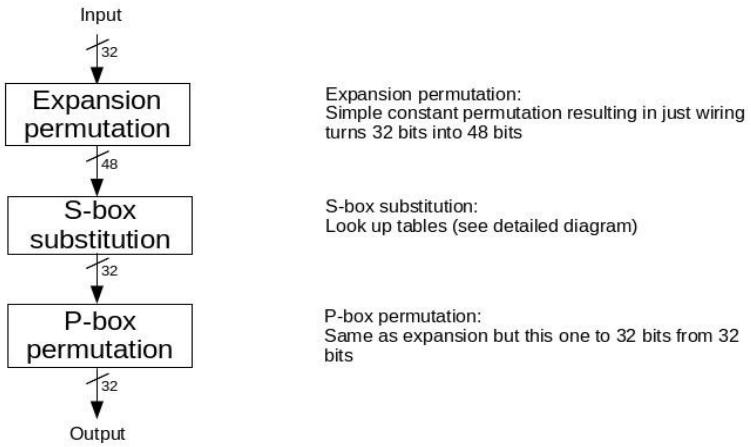


Signal Name	Type	Number of bits	Description
Input	IN	64	Input data.
K1..K16	IN	48	The generated round keys.
Output	OUT	64	Encrypted data.

Feistel

The Feistel block performs three subroutines, and the processes are explained in the following figure.

FEISTEL:

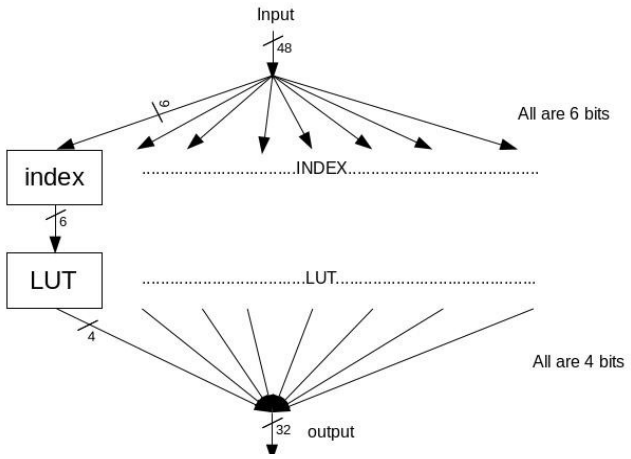


Signal Name	Type	Number of bits	Description
Input	IN	32	The output of each round (from round 1 to 16).
Output	OUT	32	The manipulated input through expansion permutation, s-box substitution, and p-box permutation.

S-Box Substitution

Find the values using a look up table, and the input as index.

S-Box:



Signal Name	Type	Number of bits	Description
Input	IN	48	The output of expansion permutation.
Output	OUT	32	The input of p-box permutation.

Requirements

In our project we will prioritize on key generation using Elliptical Curve Cryptography. The key generation algorithm of Elliptical Curve Cryptography is very secure and efficient because it uses the Private key of A, which is only known by party A. This makes it impossible for someone to create the Public key of A without being physically on party A's machine. We will also use the session key we have for triple DES encryption encrypting data we get and outputting the encrypted version. We will also try to use a small amount of external pins. We can do this by getting the input as a bitstream, outputting as a bit stream. Our project will use a total of 8 external pins. For this project we are prioritizing speed over size. We will try to minimize speed by doing computations in parallel, and minimizing the number of states in all state diagrams. Additionally, we will use Moore state machines, and make sure we do not

use large combinational blocks in order to allow us to maximize our clock speed. This design will be must faster on hardware, because on hardware the computations can be done in parallel, but on software it must be done sequentially. We will design our chip to have an area of 3mm x 3mm. We need the extra area because the multiplication algorithm used for ECC is very computationally complex on hardware. Additionally, we plan on running our chip using a clock rate of 200MHz.

Projected Timeline and Division of Tasks

Week of	Task	Member
3/23/2015	Galois Field Multiplication Galois Field Division Galois Field Addition Triple DES Encryption Point Addition Point Multiplication Controller Integrating all components Timing and area budgeting	Manish Nico Nico, Manish Nico, Manish Ed Lucas Ed, Lucas ALL ALL
3/30/2015	Galois Field Multiplication Testbench Galois Field Division Testbench Galois Field Addition Testbench Point Addition Testbench Point Multiplication Testbench Controller Testbench Integrating all components Testbench Preliminary design budget	Nico Manish Nico, Manish Ed Lucas Ed, Lucas ALL ALL
4/6/2015	Prepare for design reviews	ALL
4/13/2015	Complete Verification plan	ALL
4/20/2015	Final Verification of design	ALL
4/27/2015	Final Presentation preparation	ALL
5/4/2015	Final Report finished	ALL

Lucas is responsible for monitoring progress and negotiating task distribution.

Success Criteria

Fixed criteria:

1. (2 points) Test benches exist for all top level components and the entire design. The test benches for the entire design can be demonstrated or documented to cover all of the functional requirements given in the design specific success criteria
2. (4 points) Entire design synthesizes completely, without any inferred latches, timing arcs, and, sensitivity list warnings.
3. (2 points) Source and mapped version of the complete design behave the same for all test cases. The mapped version simulates without timing errors except at time zero.
4. (2 points) A complete IC layout is produced that passes all geometry and connectivity checks.
5. (2 points) The entire design complies with targets for area, pin count, throughput(if applicable), and clock rate. The final targets for these parameters will be determined by the course staff based on your design review. Failure to reach any of the targets will result a score of 1 out of 2 provided that you are within 50% on area, 10% on pin count, and 25% on throughput. Doing worse in any category will result in a score of 0 out of 2.

Design specific criteria:

1. Demonstrate by simulation of a Verilog test bench that the public key of A that is generated is correct.
2. Demonstrate by simulation of a Verilog test bench that the values of Skx and Sky are both generated correctly.
3. Demonstrate by simulation of a Verilog test bench that given a key, the triple DES Encryption gives out the correct values
4. Demonstrate by simulation of a Verilog test bench that the Galois field arithmetic(addition, multiplication, and division) all work correctly.