ECE 5960/6960-010: Hardware Cryptography and Security

Prepared by *Priyank Kalla*
Spring 2024, Homework # 3
Questions on Elliptic Curve Cryptography
Due Date: Wednesday March 26

In this assignment, you will be working on encipherment using asymmetric key cryptography. The first question asks you to build an ECC engine in Singular from the ground up (Galois field $\rightarrow$ Elliptic Curve $\rightarrow$ Point enumeration $\rightarrow$ Encipherment). The second question asks you to implement a point-doubling circuit using projective coordinates. Don't worry, both are only over $\mathbb{F}_8$, i.e. 3-bit fields and circuits. The designs can be done in 2 weeks without much sweat. It took me less than a week to design the HW, reverse engineer these curves and circuits, and I was also able to implement these designs, all in a few days. Just don't wait until the last minute ☺.

1) (100 points) In this question, you will design an elliptic curve crypto-cipher over a Galois field of the type $\mathbb{F}_{2^k}$. You will implement the key generation, encryption and decryption modules in Singular, and demonstrate the correct simulation. The question guides you every step of the way.

   a) Consider the finite field $\mathbb{F}_{2^3} \equiv \mathbb{F}_2[x] \pmod{P(x)}$ where $P(x) = x^3 + x^2 + 1$. Let α be a root of $P(x)$, i.e. $P(\alpha) = 0$. Note that $P(x)$ is indeed a primitive polynomial. Using Singular, enumerate the field elements $\mathbb{F}_8 = \{0, \alpha^7 = 1, \alpha, \alpha^2, \alpha^3 = \alpha^2 + 1, \alpha^4 = ?, \ldots, \alpha^6 = ?\}$. In other words, what are $\alpha^4, \alpha^5, \alpha^6$ when reduced $\pmod{P(\alpha)}$?

   b) Now consider the elliptic curve $E(\alpha^2, 1) : y^2 + xy = x^3 + \alpha^2 x^2 + 1$ over the aforementioned field $\mathbb{F}_8$. How many points are there in the elliptic curve group $G = \langle E(\alpha^2, 1), + \rangle$? Enumerate all the points on the curve, and list them as I did on the lecture slides. Use Singular to enumerate the points. *Note: be careful about* $P(x)$ *and* $E(\alpha^2, 1)$. *Both* P *and* E *are different than the ones given in the slides.*

   c) Identify a point $p(x, y)$ that acts as a *primitive element (generator)* of the aforementioned elliptic curve group G. Demonstrate that p generates all the points on E. Of course, take Singular's help to write a procedure to iterate over points $P, 2P, \ldots, nP, (n+1)P = P$, such that the least $n = |G|$.

d) Let a point $p(x_1, y_1)$ be a generator of $G = \langle E, + \rangle$. Is its inverse point p^{-1} also a generator of G? If yes, then prove it. Otherwise, give a counterexample.

e) Using the solutions to the above questions, you will now simulate El Gamal encipherment over the above elliptic curve.

- Based on Fig. 1, which is reproduced from our slides, select e_1 as a generator of $G = \langle E, + \rangle$. Select integers d, r, such that $d \neq r$.
- Select the plaintext P as a point on the elliptic curve. Compute C_1, C_2 to demonstrate encryption. [Avoid $P = (0, 1)$, as it is a trivial point on our non-supersingular curves over $\mathbb{F}_{2^k}$].
- Demonstrate decryption by re-obtaining the plaintext P.

f) **Note**: Implement the above in Singular. Please make use of "procedures" in Singular to make your code Modular. Print out the relevant parts of your computation to make it easier for me and the grader to grade it when I run your code. Attach a README to help me how to run your code. Also, in a PDF file, please describe (briefly) which points you are using as generators, what are your keys e_1, e_2, d, r and the corresponding P, C_1, C_2 values.

g) It goes without saying: feel free to borrow inspirations from the Singular files I used to give you a demo of ECC in class; those Singular files are uploaded on Canvas: ecc-f8-example.sing.

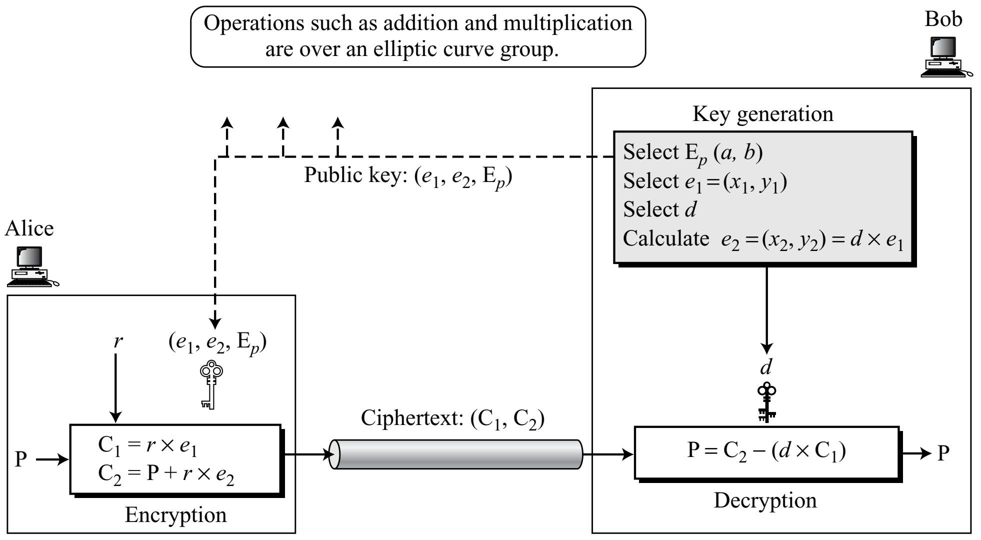
Operations such as addition and multiplication are over an elliptic curve group.
Bob
Key generation
Select $E_p(a, b)$
Select $e_1 = (x_1, y_1)$
Select d
Calculate $e_2 = (x_2, y_2) = d \times e_1$
Public key: (e_1, e_2, E_p)
Alice
r
(e_1, e_2, E_p)
d
P
$C_1 = r \times e_1$
$C_2 = P + r \times e_2$
Encryption
Ciphertext: (C_1, C_2)
$P = C_2 - (d \times C_1)$
P
Decryption

Fig. 1: El Gamal over ECC

2) (100 points) In this question, you will design a *digital logic circuit* that performs *point doubling* $R = 2P$ operation (not point addition!) over elliptic curves using the *projective coordinate* system. You will first design (or re-use from HW 2) a multiplier circuit, use it as a building block to perform point doubling. You will implement your design in Verilog or VHDL,

and demonstrate that point addition is being performed correctly. Proceed as follows:

a) We will use the same finite field as in the previous question: $\mathbb{F}_8 \equiv \mathbb{F}_2[x] \pmod{P(x) = x^3 + x^2 + 1}$ with $P(\alpha) = 0$. Denote the degree of $P(x)$ as k; of course, here $k = 3$.

b) Design a $k = 3$ bit *finite field multiplier* that takes $A = \{a_2, a_1, a_0\}$ and $B = \{b_2, b_1, b_0\}$ as 3-bit inputs, and produces $Z = \{z_2, z_1, z_0\}$ as a 3-bit output. Note that we will have:

$$A = a_0 + a_1\alpha + a_2\alpha^2 \quad (1)$$

$$B = b_0 + b_1\alpha + b_2\alpha^2 \quad (2)$$

$$Z = z_0 + z_1\alpha + z_2\alpha^2 \quad (3)$$

Such that $Z = A \cdot B \pmod{P(\alpha)}$. Of course, you have already designed 2 multipliers in the last HW (Mastrovito and Montgomery). Just pick whichever one you like. Also, please double check that the primitive polynomial that you used in the design of HW 2 was indeed $P(x) = x^3 + x^2 + 1$.

c) Implement the design in Verilog/VHDL (`GFMult(A, B, Z)` module) and demonstrate/simulate using a testbench the following input-output combinations:

- $A = (0, 1, 0) = \alpha, B = (1, 0, 0) = \alpha^2, Z = (1, 0, 1) = \alpha^2 + 1$
- $A = \alpha^2 + 1, B = \alpha^2 + \alpha + 1, Z = ?$

d) Using your `GFMult` module, create a squarer module by connecting $A = B$ inputs; call it the `GFSQR` module.

e) Design a `GFADD(A, B, Z)` Verilog Module, such that $Z = A+B$ over $\mathbb{F}_8$. [Remember, addition in Galois Fields is just a bit-wise XOR].

f) In the lecture slides (ECC-GF.pdf), I have given you the correct formulas for point addition and doubling operations. Implement a Verilog Module to perform **point doubling over projective coordinates**. Your `PointDouble(X3,Y3,Z3,X1,Y1,Z1)` Verilog/VHDL module should instantiate `GFADD, GFMult, GFSQR` modules accordingly to compute each of the 3-bit $X3, Y3, Z3$ outputs.

g) Draw a Data Flow Graph (DFG) for $X3, Y3, Z3$, using the 3 operators, to show how your adders, multipliers and squarers are organized.

h) Demonstrate that your `PointDouble()` module correctly computes the doubling of the following affine points:

- Pick $Z1 = 1$ to keep computations simple. Note that since each coordinate of a point is in F_8, each of $X1, Y1, Z1$ is a 3-bit vector.

- For affine point $P = (\alpha, 1)$, simulate $2P$ on your Verilog Testbench. What is $2P$?
- For affine point $P = (\alpha^3, \alpha + 1)$, simulate $2P$ on your Verilog Testbench. What is $2P$ for this case?
- Note that (X_1, Y_1, Z_1) computed by your circuit is actually $(X_1/Z_1, Y_1/Z_1)$ in the affine space! You can of course check your answer with Singular.