Homework Assignment # 3

Miguel Gomez U1318856

2024-03-25 14:55:16

Contents

1 HW3 - Problem 1

design an elliptic curve crypto-cipher over a Galois field of the type $\mathbb{F}_{2^k}$. Implement the key generation, encryption and decryption modules in Singular, and demonstrate the correct simulation.

1.1 a) what are $\alpha^4, \alpha^5, \alpha^6$ when reduced (mod $P(\alpha)$)

Consider the finite field $\mathbb{F}_{2^k} \equiv \mathbb{F}_2[x]$ (mod $P(x)$) 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 $F_8 = 0, \alpha^7 = 1, \alpha, \alpha^2, \alpha^3 = \alpha^2 + 1, \alpha^4 = ?, ..., \alpha^6 = ?$. In other words, what are $\alpha^4, \alpha^5, \alpha^6$ when reduced (mod $P(\alpha)$)?

```
start=$(date +%s.%N)
Singular ./sing/point_enumeration.sing | grep -v -e "\*\* loaded\|\*\* library"
end=$(date +%s.%N)
echo "Execution Time: $(echo "$end - $start" | bc) seconds"
```

As can be seen in the output below, $\alpha^4,\ \alpha^5,\ \alpha^6$ are as follows:

$$\alpha^4 = \alpha^2 + \alpha + 1$$
$$\alpha^5 = \alpha + 1$$
$$\alpha^6 = \alpha^2 + \alpha$$

1.1.1 output of point-enumeration.sing results

```
                     SINGULAR                                 /  Development
 A Computer Algebra System for Polynomial Computations       /   version 4.3.2
                                                           0<
 by: W. Decker, G.-M. Greuel, G. Pfister, H. Schoenemann     \   Feb 2023
FB Mathematik der Universitaet, D-67653 Kaiserslautern        \
// ** executing /home/speedy/repos/singular/git/Singular/Singular/Singular/.libs/../LIB/.singularrc
// ** but for functionality you may wish to change to the new
// ** format. Please refer to the manual for further information.
================================
when x = 0
poly f is:
y2+1
poly f factorizes as follows:
[1]:
   _[1]=1
   _[2]=y+1
[2]:
   1,2
================================
when x = A^0, : x = 1
P1(x,y) = (1,y2+(A2))
================================
when x = A^1, : x = (A)
P1(x,y) = ((A),1)
P2(x,y) = ((A),y+(A+1))
================================
when x = A^2, : x = (A2)
P1(x,y) = ((A2),(A))
P2(x,y) = ((A2),y+(A2+A))
================================
when x = A^3, : x = (A2+1)
P1(x,y) = ((A2+1),(A+1))
P2(x,y) = ((A2+1),y+(A2+A))
================================
when x = A^4, : x = (A2+A+1)
P1(x,y) = ((A2+A+1),(A))
P2(x,y) = ((A2+A+1),y+(A2+1))
================================
when x = A^5, : x = (A+1)
P1(x,y) = ((A+1),0)
P2(x,y) = ((A+1),y+(A+1))
================================
when x = A^6, : x = (A2+A)
P1(x,y) = ((A2+A),1)
P2(x,y) = ((A2+A),y+(A2+A+1))
================================
when x = A^7, : x = 1
P1(x,y) = (1,y2+(A2))
================================
Auf Wiedersehen.
Execution Time: .060719112 seconds
```

1.2 b) Enumerate all the points on the curve

```
start=$(date +%s.%N)
Singular ./sing/point_gen.sing | grep -v -e "\*\* loaded\|\*\* library"
end=$(date +%s.%N)
echo "Execution Time: $(echo "$end - $start" | bc) seconds"
```

1.2.1 output of point_gen.sing results

```
                     SINGULAR                                 /  Development
 A Computer Algebra System for Polynomial Computations       /   version 4.3.2
                                                           0<
 by: W. Decker, G.-M. Greuel, G. Pfister, H. Schoenemann     \   Feb 2023
FB Mathematik der Universitaet, D-67653 Kaiserslautern        \
// ** executing /home/speedy/repos/singular/git/Singular/Singular/Singular/.libs/../LIB/.singularrc
// ** but for functionality you may wish to change to the new
// ** format. Please refer to the manual for further information.
P = ((A2+1), (A+1))
Calling doubleP on P
received/ 2P:
2P = ((A2), (A2+A))
3P = ((A2+A), (A2+A+1))
4P = ((A), (A+1))
5P = ((A+1), 0)
6P = ((A2+A+1), (A2+1))
7P = (0, 1)
8P = ((A2+A+1), (A))
9P = ((A+1), (A+1))
10P = ((A), 1)
11P = ((A2+A), 1)
12P = ((A2), (A))
13P = ((A2+1), (A2+A))
14P = (0, 0)
15P = ((A), (A2+A+1))
16P = ((A2+1), (A2+A))
Auf Wiedersehen.
Execution Time: .074262521 seconds
```

Shown above is the enumeration of the points and below they are added with nicer formatting.

$$
\begin{aligned}
P &= (\alpha^2+1, \alpha+1) \\
2P &= (\alpha^2, \alpha^2+\alpha) \\
3P &= (\alpha^2+\alpha, \alpha^2+\alpha+1) \\
4P &= (\alpha, \alpha+1) \\
5P &= (\alpha+1, 0) \\
6P &= (\alpha^2+\alpha+1, \alpha^2+1) \\
7P &= (0, 1) \\
8P &= (\alpha^2+\alpha+1, \alpha) \\
9P &= (\alpha+1, \alpha+1) \\
10P &= (\alpha, 1) \\
11P &= (\alpha^2+\alpha, 1) \\
12P &= (\alpha^2, \alpha) \\
13P &= (\alpha^2+1, \alpha^2+\alpha) \\
14P &= (0, 0)
\end{aligned}
$$

1.3 c) Identify a point $P(x, y)$ that acts as a primitive element

This result above provided by the Singular procedure proves that we can generate the points on the curve as well as proving that our starting point below is a generator.

$$P = (\alpha^3, \alpha^5) = (\alpha^2 + 1, \alpha + 1)$$

1.4 d) Is its inverse point P^{-1} also a generator of G?

Let a point $P(x1, y1)$ 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.

```
start=$(date +%s.%N)
Singular ./sing/point_inversion.sing | grep -v -e "\*\* loaded\|\*\* library"
end=$(date +%s.%N)
echo "Execution Time: $(echo "$end - $start" | bc) seconds"
```

1.4.1 output of inversion results

```
                     SINGULAR                                 /  Development
 A Computer Algebra System for Polynomial Computations       /   version 4.2.1
                                                           0<
 by: W. Decker, G.-M. Greuel, G. Pfister, H. Schoenemann     \   May 2021
FB Mathematik der Universitaet, D-67653 Kaiserslautern        \  Debian 1:4.2.1-p3+ds-1
// ** but for functionality you may wish to change to the new
// ** format. Please refer to the manual for further information.
printing 1P:
((A2+1), (A+1))
printing 2P:
((A2), (A2+A))
printing 3P:
((A2+A), (A2+A+1))
printing 4P:
((A), (A+1))
printing 5P:
((A+1), 0)
printing 6P:
((A2+A+1), (A2+1))
printing 7P:
(0, 1)
printing 8P:
((A2+A+1), (A))
printing 9P:
((A+1), (A+1))
printing 10P:
((A), 1)
printing 11P:
((A2+A), 1)
printing 12P:
((A2), (A))
printing 13P:
((A2+1), (A2+A))
printing 14P:
(0, 0)
Auf Wiedersehen.
Execution Time: .009570786 seconds
```

1.5 e) simulate El Gamal encipherment

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

1.5.1 selecting parameters for encryption

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 $d \neq r$.

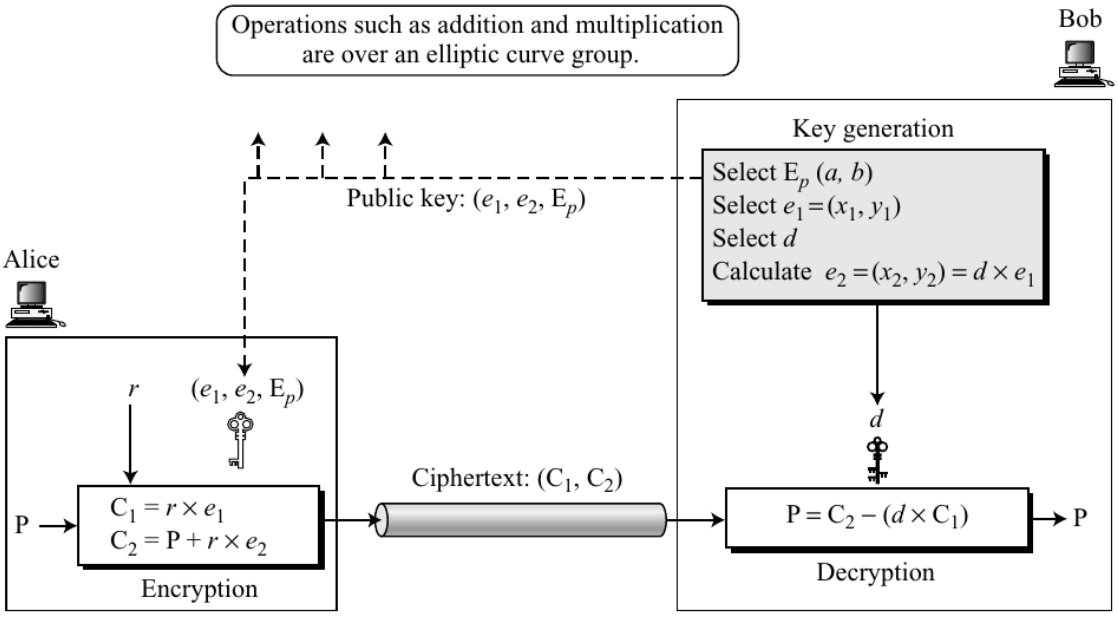

Figure 1: El Gamal over ECC

1.5.2 Selecting plain-text

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 $\mathbb{F}_{2^k}$].

1.5.3 Demonstrate decryption by re-obtaining the plaintext P

1.6 f) Notes:

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 understand 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.

1.7 g) Permissions for using skeleton provided

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

2 Problem 2

In this question, you will design a digital logic circuit that performs point doubling $R = 2P$ (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 doubling. You will implement your design in Verilog or VHDL, and demonstrate that point addition is being performed correctly.

2.1 a) Defining field and terms

We will use the same finite field as in the previous question: F8 F2 [x] (mod P(x) =x3 + x2 + 1) with P() = 0. Denote the degree of P(x) as k; of course, here k = 3.

2.2 b) Design multiplier

Design a k = 3 bit finite field multiplier that takes A = {a2, a1 , a0 } and B = {b2 , b1 , b0 } as 3-bit inputs, and produces Z = {z2 , z1 , z0} as a 3-bit output. Note that we will have:

$$A = a_0 + a_1\alpha + a_2\alpha^2$$
$$B = b_0 + b_1\alpha + b_2\alpha^2$$
$$Z = z_0 + z_1\alpha + z_2\alpha^2$$

Such that $Z = A \cdot B \bmod P(\alpha))$. 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 $P(x) = x^3 + x^2 + 1$.

2.3 c) Implementation in Verilog

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

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

2.3.1 2.

$$A = \alpha^2 + 1$$
$$B = \alpha^2 + \alpha + 1$$
$$Z = ?$$

2.4 d) Design Squarer

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

2.5 e) Design GFADD

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].

2.6 f) Implement Point doubling in projective coordinates

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($X_3, Y_3, Z_3, X_1, Y_1, Z_1$) Verilog/VHDL module should instan- tiate GFADD, GFMult, GFSQR modules accordingly to compute each of the 3-bit X_3, Y_3, Z_3 outputs.

2.7 g) DFG

Draw a Data Flow Graph (DFG) for X_3, Y_3, Z_3, using the 3 operators, to show how your adders, multipliers and squarers are organized.

2.8 h) Simulation and example demonstrations

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

2.8.1 Defining projective plane

Pick $Z_1 = 1$ to keep computations simple. Note that since each coordinate of a point is in $\mathbb{F}_8$, each of X_1, Y_1, Z_1 is a 3-bit vector.

2.8.2 $P = (\alpha, 1)$ simulate $2P$

For affine point $P = (\alpha, 1)$, simulate $2P$ on your Verilog Testbench. What is $2P$?

2.8.3 $P = (\alpha^3, \alpha + 1)$ simulate $2P$

For affine point $P = (\alpha^3, \alpha + 1)$, simulate $2P$ on your Verilog Testbench. What is $2P$ for this case?

2.8.4 Notes:

Note that (X_1, Y_1, Z_1) computed by your circuit is actually $(\frac{X_1}{Z_1}, \frac{Y_1}{Z_1})$ in the affine space! You can of course check your answer with Singular.