CO685 (The Mathematics of Public Key Cryptography) - Course Project

Point Multiplication on Elliptic Curves

Results and Examples

Vishnu V Narayan, M.Math. C&O, 20638622

1. **Miller-Rabin Strong Pseudoprime Test (Testing `BI_Miller_Rabin`)**

   Compile and execute file: Miller_Test.cpp

   Output:

```
C:\Users\VVN\Desktop\Assignments\F15\CO685\Project>g++ Miller_Test.cpp
BigInteger.cpp -o Miller_Test.exe

C:\Users\VVN\Desktop\Assignments\F15\CO685\Project>Miller_Test.exe
671998030559713968361666935769          1
786723452198609412836798763452 4        0
786914378917592384751987982741          0
327894769823456999348573946938457       0
897621453374831912718274341 77          0
282174488599599500573849980909          1
521419622856657689423872613771          1
234110404094782374117878274 81          0
362736035870515331128527330659          1
```

   Result: Correct (verified against Wolfram|Alpha).

   Comments: The 4 primes (with result 1 above) are the first 4 given primes of length 30 digits from the list of large primes here: https://primes.utm.edu/lists/small/small.html. The other numbers were randomly generated.

2. **Elliptic Curve Point Addition Test (Testing `EC_Add` and `EC_Double`)**

   - **Compile and execute file**: Add_Test_1.cpp

     Output:

```
C:\Users\VVN\Desktop\Assignments\F15\CO685\Project>g++ Add_Test_1.cpp
BigInteger.cpp ECPO.cpp -o Add_Test_1.exe

C:\Users\VVN\Desktop\Assignments\F15\CO685\Project>Add_Test_1.exe
O       1, 5    1, 8    2, 3    2, 10   9, 6    9, 7    12, 2   12, 11
1, 5    2, 10   O       1, 8    9, 7    2, 3    12, 2   12, 11  9, 6
1, 8    O       2, 3    9, 6    1, 5    12, 11  2, 10   9, 7    12, 2
2, 3    1, 8    9, 6    12, 11  O       12, 2   1, 5    2, 10   9, 7
2, 10   9, 7    1, 5    O       12, 2   1, 8    12, 11  9, 6    2, 3
9, 6    2, 3    12, 11  12, 2   1, 8    9, 7    O       1, 5    2, 10
9, 7    12, 2   2, 10   1, 5    12, 11  O       9, 6    2, 3    1, 8
12, 2   12, 11  9, 7    2, 10   9, 6    1, 5    2, 3    1, 8    O
12, 11  9, 6    12, 2   9, 7    2, 3    2, 10   1, 8    O       1, 5
```

     Result: Correct (verified against the textbook, Table 6.1).

     Comments: The example is from the textbook, and is provided below for comparison.

	$\mathcal{O}$	$(1,5)$	$(1,8)$	$(2,3)$	$(2,10)$	$(9,6)$	$(9,7)$	$(12,2)$	$(12,11)$
$\mathcal{O}$	$\mathcal{O}$	$(1,5)$	$(1,8)$	$(2,3)$	$(2,10)$	$(9,6)$	$(9,7)$	$(12,2)$	$(12,11)$
$(1,5)$	$(1,5)$	$(2,10)$	$\mathcal{O}$	$(1,8)$	$(9,7)$	$(2,3)$	$(12,2)$	$(12,11)$	$(9,6)$
$(1,8)$	$(1,8)$	$\mathcal{O}$	$(2,3)$	$(9,6)$	$(1,5)$	$(12,11)$	$(2,10)$	$(9,7)$	$(12,2)$
$(2,3)$	$(2,3)$	$(1,8)$	$(9,6)$	$(12,11)$	$\mathcal{O}$	$(12,2)$	$(1,5)$	$(2,10)$	$(9,7)$
$(2,10)$	$(2,10)$	$(9,7)$	$(1,5)$	$\mathcal{O}$	$(12,2)$	$(1,8)$	$(12,11)$	$(9,6)$	$(2,3)$
$(9,6)$	$(9,6)$	$(2,3)$	$(12,11)$	$(12,2)$	$(1,8)$	$(9,7)$	$\mathcal{O}$	$(1,5)$	$(2,10)$
$(9,7)$	$(9,7)$	$(12,2)$	$(2,10)$	$(1,5)$	$(12,11)$	$\mathcal{O}$	$(9,6)$	$(2,3)$	$(1,8)$
$(12,2)$	$(12,2)$	$(12,11)$	$(9,7)$	$(2,10)$	$(9,6)$	$(1,5)$	$(2,3)$	$(1,8)$	$\mathcal{O}$
$(12,11)$	$(12,11)$	$(9,6)$	$(12,2)$	$(9,7)$	$(2,3)$	$(2,10)$	$(1,8)$	$\mathcal{O}$	$(1,5)$

Table 6.1: Addition table for $E : Y^2 = X^3 + 3X + 8$ over $\mathbb{F}_{13}$

- **Compile and execute file**: Add_Test_2.cpp
  Output:

```
O        0, 0    1, 5    1, 18   9, 5    9, 18   11, 10  11, 13
0, 0     O       1, 18   1, 5    18, 13  18, 10  21, 6   21, 17
1, 5     1, 18   0, 0    O       13, 18  16, 8   17, 10  19, 22
1, 18    1, 5    O       0, 0    16, 15  13, 5   19, 1   17, 13
9, 5     18, 13  13, 18  16, 15  18, 10  O       15, 3   19, 1
9, 18    18, 10  16, 8   13, 5   O       18, 13  19, 22  15, 20
11, 10   21, 6   17, 10  19, 1   15, 3   19, 22  13, 18  O
11, 13   21, 17  19, 22  17, 13  19, 1   15, 20  O       13, 5
13, 5    16, 8   9, 18   18, 10  1, 18   13, 18  11, 13  15, 3
13, 18   16, 15  18, 13  9, 5    13, 5   1, 5    15, 20  11, 10
15, 3    20, 19  15, 20  20, 4   17, 13  11, 10  13, 5   9, 5
15, 20   20, 4   20, 19  15, 3   11, 13  17, 10  9, 18   13, 18
16, 8    13, 5   18, 10  9, 18   1, 5    16, 15  21, 17  20, 19
16, 15   13, 18  9, 5    18, 13  16, 8   1, 18   20, 4   21, 6
17, 10   19, 1   21, 6   11, 10  15, 20  21, 17  18, 13  1, 5
17, 13   19, 22  11, 13  21, 17  21, 6   15, 3   1, 18   18, 10
18, 10   9, 18   13, 5   16, 8   0, 0    9, 5    17, 13  20, 4
18, 13   9, 5    16, 15  13, 18  9, 18   0, 0    20, 19  17, 10
19, 1    17, 10  11, 10  21, 6   20, 4   11, 13  9, 5    1, 18
19, 22   17, 13  21, 17  11, 13  11, 10  20, 19  1, 5    9, 18
20, 4    15, 20  15, 3   20, 19  21, 17  19, 1   18, 10  16, 15
20, 19   15, 3   20, 4   15, 20  19, 22  21, 6   16, 8   18, 13
21, 6    11, 10  19, 1   17, 10  20, 19  17, 13  16, 15  0, 0
21, 17   11, 13  17, 13  19, 22  17, 10  20, 4   0, 0    16, 8
```

  Result: Correct (Verified by hand-checking randomly chosen outputs, and checking that every result point is on the curve) – the result has been truncated to the first 8 columns.
  Comments: The example is from here: https://www.certicom.com/index.php/31-example-of-an-elliptic-curve-group-over-fp and is the elliptic curve E: $y^2 = x^3 + x$ over the field F_{23}.

3. **Elliptic Curve Point Multiplication Test (Testing `EC_Basic_Multiply`, `EC_Double_and_Add` and `EC_WindowNAF`)**
   - **Compile and execute file**: Point_Multiply_1.cpp
     Output:

```
C:\Users\VVN\Desktop\Assignments\F15\CO685\Project>g++
Point_Multiply_1.cpp BigInteger.cpp ECPO.cpp -o pm1.exe

C:\Users\VVN\Desktop\Assignments\F15\CO685\Project>pm1.exe


_____________________________________________
Point P1 results : 9 times 1, 5
_____________________________________________
Basic Multiply result :         O
Double and Add result :         O
WindowNAF result :              O


_____________________________________________
Point P2 results : 5 times 2, 3
_____________________________________________
Basic Multiply result :         1, 8
Double and Add result :         1, 8
WindowNAF result :              1, 8
```

     Result: Correct (Verified from Table 6.1)
     Comments: All algorithms return the same results.

   - **Compile and execute file**: Point_Multiply_2.cpp
     Output:

```
C:\Users\VVN\Desktop\Assignments\F15\CO685\Project>g++
Point_Multiply_2.cpp BigInteger.cpp ECPO.cpp -o pm2.exe

C:\Users\VVN\Desktop\Assignments\F15\CO685\Project>pm2.exe


_____________________________________________
Point P1 results
_____________________________________________
Double and Add running time :           4.848
WindowNAF running time :                3.934

Double and Add result :
195180093019200700837629478134,204575605965933212464535926849
WindowNAF result :
195180093019200700837629478134,204575605965933212464535926849



_____________________________________________
Point P2 results
_____________________________________________
```

```
Double and Add running time :           4.908
WindowNAF running time :                4.005

Double and Add result :
351451254754743216179481608682,248955259375756947649279424378
WindowNAF result :
351451254754743216179481608682,248955259375756947649279424378


_____________________________________________
Point P3 results
_____________________________________________
Double and Add running time :           4.897
WindowNAF running time :                3.969

Double and Add result :         101345269151241334823602179763,
143430974503486223797240268621
WindowNAF result :              101345269151241334823602179763,
143430974503486223797240268621
```

Result: Correct (algorithms return same result, performance is as expected).
Comments: Both algorithms return the same results. EC_Basic_Multiply could not participate in this test as it would take far too long to finish execution. EC_WindowNAF is non-negligibly faster than EC_Double_and_Add.