Using pairing friendly MCL library via Python Wrapper

Łukasz Krzywiecki
Department of Fundamentals of Computer Science
Wrocław University of Science Technology
Wrocław, Poland
lukasz.krzywiecki@pwr.edu.pl

1 Introduction

To install MCL library please follow instructions provided on URL:
https://github.com/herumi/mcl
In the rest of this tutorial I present instructions I run on my MacBook with macOS Monterey.

2 GIT cloning and compiling

My MCL library is: /mcl-org located in my home directory.

```
git clone https://github.com/herumi/mcl
cd mcl-org
make -j4
```

3 Test compilation and execution

To compile bls12_test:

```
make bin/bls12_test.exe && bin/bls12_test.exe
```

To run bls12_test test:

```
MacBook-Pro:mcl-org lukasz$ cd bin
MacBook-Pro:bin lukasz$ ls
bls12_test.exe
MacBook-Pro:bin lukasz$ ./bls12_test.exe
```

```
JIT 1
ctest:module=size
ctest:module=naive
i=0 curve=BLS12_381
testMulSmall
testFp2Dbl_mul_xi1
testMisc1
testMul2
G1
G2
GT
G1::mulCT       496.244Kclk
G1::mul         500.207Kclk
G1::add           1.315Kclk
G1::dbl           1.065Kclk
G2::mulCT         1.215Mclk
G2::mul           1.227Mclk
G2::add           3.634Kclk
G2::dbl           2.294Kclk
GT::pow           2.508Mclk
G1::setStr chk 177.555Kclk
G1::setStr        5.070Kclk
G2::setStr chk 189.399Kclk
G2::setStr       10.117Kclk
hashAndMapToG1 369.590Kclk
hashAndMapToG2 692.429Kclk
Fp::add          16.40 clk
Fp::sub          17.23 clk
Fp::add 2        17.84 clk
Fp::mul2         16.09 clk
Fp::mulSmall8    59.69 clk
Fp::mulUnit8     54.14 clk
Fp::mul9         56.84 clk
Fp::mulUnit9     69.28 clk
Fp::neg          10.26 clk
Fp::mul         108.39 clk
Fp::sqr         104.77 clk
Fp::inv          13.672Kclk
Fp::pow          54.914Kclk
Fr::add          12.99 clk
Fr::sub          11.45 clk
Fr::neg           8.99 clk
Fr::add 2        11.54 clk
Fr::mul2         13.11 clk
Fr::mul          64.66 clk
Fr::sqr          61.34 clk
Fr::inv           6.583Kclk
Fr::pow          21.383Kclk
Fp2::add         24.84 clk
Fp2::sub         26.07 clk
Fp2::neg         17.06 clk
Fp2::mul2        22.80 clk
Fp2::mul        325.48 clk
Fp2::mul_xi      27.46 clk
Fp2::sqr        250.73 clk
Fp2::inv         14.117Kclk
FpDbl::addPre    11.80 clk
```

```
FpDbl::subPre    12.61 clk
FpDbl::add       20.76 clk
FpDbl::sub       20.97 clk
FpDbl::mulPre    52.54 clk
FpDbl::sqrPre    43.46 clk
FpDbl::mod       70.77 clk
Fp2Dbl::mulPre  199.07 clk
Fp2Dbl::sqrPre  128.58 clk
GT::add        155.99 clk
GT::mul          6.578Kclk
GT::sqr          4.795Kclk
GT::inv         18.732Kclk
pairing          2.108Mclk
millerLoop     882.112Kclk
finalExp         1.248Mclk
precomputeG2   527.152Kclk
precomputedML  678.735Kclk
millerLoopVec    4.517Mclk
ctest:module=finalExp
finalExp    1.231Mclk
ctest:module=mul_012
ctest:module=pairing
ctest:module=multi
BN254
calcBN1   92.511Kclk
naiveG2   25.406Kclk
calcBN2 195.192Kclk
naiveG2 120.549Kclk
BLS12_381
calcBN1 159.245Kclk
naiveG1  59.472Kclk
calcBN2 369.981Kclk
naiveG2 230.742Kclk
ctest:module=deserialize
verifyOrder(1)
deserializeG1 292.935Kclk
deserializeG2 426.942Kclk
verifyOrder(0)
deserializeG1 115.676Kclk
deserializeG2 253.788Kclk
ctest:module=verifyG1
ctest:module=verifyG2
ctest:name=bls12_test, module=9, total=4652, ok=4652, ng=0,
    exception=0
```

4 Python Wrapper

To install a Python wrapper please go for:

`https://github.com/umberto10/mcl-python`

and follow the instructions. This repository, build for the version of MCL (as of June 2022), is created by my student Hubert Mazur, as an improved version

of a previous wrapper by another my student Piotr Szyma:
`https://github.com/piotrszyma/mcl-python`.
My Python wrapper for MCL library, cloned from:
`https://github.com/umberto10/mcl-python`
is located in: `/mcl-python` in my home directory.
Then, according to instructions, I modified the file:
`/Users/lukasz/mcl-python/mcl/hook.py`
by setting the line:

```
...
DIR_FOR_LINKER = os.environ.get("MCL_PATH", "/Users/lukasz/mcl-
    org/")
...
```

5 Sample Python code

Now in my Python source code I add the following line

```
import sys
sys.path.insert(1, '/Users/lukasz/mcl-python')
...
...
```

We are going to use the asymmetric pairing function e:

$$e : G_1 \times G_2 \to G_T$$

with the usual property:

$$e(g_1^a, g_2^b) = e(g_1, g_2^b)^a = e(g_1^a, g_2)^b = e(g_1, g_2)^{ab}.$$

Thus we need to import the appropriate groups.
Note that `Fr` in the code below is used for exponents a and b.:

```
import sys
sys.path.insert(1, '/Users/lukasz/mcl-python')

from mcl import GT
from mcl import G2
from mcl import G1
from mcl import Fr
```

In the snippets of code below you can find the fundamental operations used for the cryptographic scheme implementations. Declaring variables for exponents:

```
a = Fr()
b = Fr()
c = Fr()
d = Fr()
k = Fr()
```

Setting exponents to random values:

```
a = Fr()
b = Fr()
c = Fr()
```

Setting exponents to random values:

```
a.setByCSPRNG()
b.setByCSPRNG()
c.setByCSPRNG()
```

Setting exponents to integer values:

```
a.setInt(2)
b.setInt(3)
c.setInt(6)
```

New generators from hash values:

```
h1 = G1()
h1 = G1.hashAndMapTo(b"abcd")
h2 = G2()
h2 = G2.hashAndMapTo(b"efgh")
```

k exponent from hash value:

```
k = Fr.setHashOf(b"abcd")
```

Use of pairing functions:

```
e1 = GT.pairing(h1 * a, h2 *b)
e2 = GT.pairing(g1 * a, g2 *b)
```

5.1 A more complex code

Note that the MCL source code (and the Python wrapper) use a typical elliptic curve notations for operations. We have *multiplications* instead of *exponentiations* in groups G_1 and G_2. Thus for the pairing function we have the property:

$$e(g_1 \cdot a, g_2 \cdot b) = e(g_1, g_2 \cdot b)^a = e(g_1 \cdot a, g_2)^b = e(g_1, g_2)^{ab}.$$

However, the *exponentiation* still remains in G_T.
Please check a more complex code:

```
t = time.time()

a.setByCSPRNG()
b.setByCSPRNG()
c.setByCSPRNG()

g1 = G1.hashAndMapTo(b"abcd")
g2 = G2.hashAndMapTo(b"abcd")

h1 = g1 * a * b

k2 = g2 * c

e1 = GT.pairing(h1, k2)
e2 = GT.pairing(g1, k2 * a * b)

print("uwaga")
print(e1 == e2)

a.setInt(2)
b.setInt(3)
c.setInt(6)
v.setInt(5)

d.setByCSPRNG()

h1 = g1 * d
k1 = h1 - g1
z2 = g2 * d - g2

e7 = GT.pairing(k1, g2)
e8 = GT.pairing(g1, z2)
print(e8 == e7)



h1 = g1 * d
k1 = h1 - g1
z2 = g2 * c - g2

e2 = GT.pairing(g1, g2)
e8 = GT.pairing(g1, z2)
e5 = e2 * e2 * e2 * e2 * e2
ee1 = e2 ** v

print("exp check")
```

```
   print(ee1 == e5)
48
   print (time.time()-t)
50
   e3 = e2 * e2 * e2 * e2 * e2
52
   n.setByCSPRNG()
54
   Fr.setInt(n, 5)
56
   e9 = e2 ** n
58 print(e9 == e3)

60 p = G1.hashAndMapTo(b"abcd")
   q = G2.hashAndMapTo(b"abcd")
62
   p1 = p * b
64 q1 = q * b

66 e2 = GT.pairing(p, q)
   e3 = GT.pairing(p1, q1)
68 e4 = GT.pairing(p1, q1)

70 f.setInt(9)
   e5 = e2 ** f
72
   print(e4 == e3)
74 d.setInt(6)

76

78
   e6 = e2 * e2 * e2 * e2 * e2 * e2
80 e7 = e2 ** d

82 print(e6 == e7)
```

The results of the above source code, starting from the line no 17. The code is run in the `PyCharm` IDE:

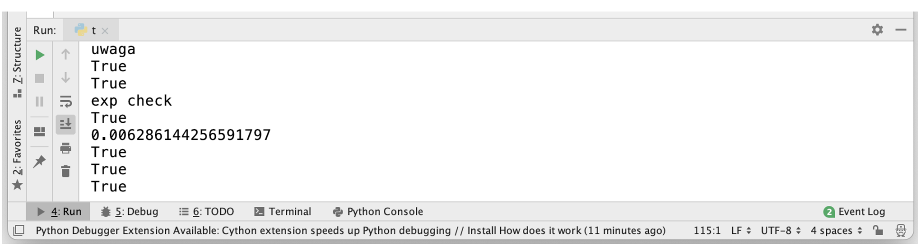

Figure 1: Results of the *more complex code* in `PyCharm` IDE.

6 Working with Debugger in `PyCharm` IDE

Debugger just runs smoothly...

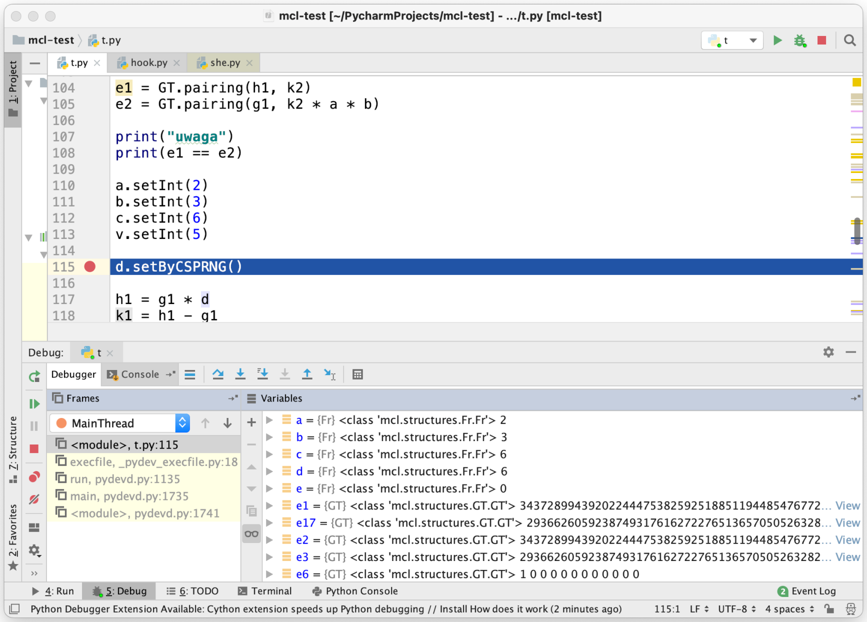

Figure 2: Debugging session in `PyCharm` IDE.