



GNU Multi-Precision Library



密碼學與應用
海洋大學資訊工程系
丁培毅

GMP 6.2.0 Installation

✧ gcc/g++

- ✧ Source download: <https://gmplib.org/>
- ✧ complete build: configure, make, make check, runbench
- ✧ `include\gmp.h, gmpxx.h`
`lib\libgmp.a, libgmp.la, libgmpxx.a, libgmpxx.la`
⇒ C:\Progra~2\dev-cpp\MinGW64\
- ✧ <https://gmplib.org/manual/>

✧ python

- ✧ `conda install -c conda-forge gmpy2`
- ✧ <https://gmpy2.readthedocs.io/en/latest>

g++ GMP examples

```
// g++ testgmp.cpp -lgmp -o testgmp.exe
#include <iostream>
#include <cstdio>
#include <gmp.h>
int main(void) {
    std::ios::sync_with_stdio();
    mpz_t dataout, base;

    mpz_inits(dataout, base, NULL);
    mpz_set_str(base, "2", 10);
    mpz_pow_ui(dataout, base, 223);
    mpz_sub_ui(dataout, dataout, 1);
    mpz_out_str(stdout, 10, dataout);
    std::cout << std::endl;
    return 0;
}
```

GMP Integer Functions

- ✧ Initializing: `mpz_init(mpz_t)`, `mpz_inits(mpz_t, ..., NULL)`
- ✧ Cleaning: `mpz_clear(mpz_t)`, `mpz_clears(mpz_t, ..., NULL)`
- ✧ Assigning: `mpz_set()`, `mpz_set_ui()`, `mpz_set_str()`
- ✧ Init & assign: `mpz_init_set()`, `mpz_init_set_{ui, si, d, str}()`
- ✧ Arithmetic: `mpz_{add,sub,mul,addmul,submul,neg,abs}()`
- ✧ Division: `mpz_{cdiv,fdiv,tdiv}_{q,r,qr}()`, `mpz_divisible_p()`
- ✧ Exponentiation: `mpz_powm`
- ✧ Roots: `mpz_root()`, `mpz_sqrt()`
- ✧ Number Theoretic Functions: `mpz_probab_prime_p()`,
`mpz_{nextprime,gcd,gcdext,lcm,invert,jacobi,legendre}()`
- ✧ Comparisons: `mpz_{cmp,cmpabs,sgn}()`
- ✧ Random Numbers: `gmp_randinit()`, `mpz_urandomm()`
- ✧ Input/Output: `mpz{inp,out}_str()`

Textbook RSA

```
void rsa_keys(mpz_t n, mpz_t d,  
               const mpz_t p, const mpz_t q, const mpz_t e) {  
    mpz_mul(n, p, q);  
  
    mpz_t p_1, q_1, lambda, gcd, mul, mod;  
    mpz_inits(p_1, q_1, lambda, gcd, mul, mod, NULL);  
    mpz_sub_ui(p_1, p, 1);  
    mpz_sub_ui(q_1, q, 1);  
    mpz_lcm(lambda, p_1, q_1);  
    //printf("lambda = %s\n", mpz_get_str(NULL, 0, lambda));  
  
    mpz_gcd(gcd, e, lambda);  
    assert(mpz_cmp_ui(gcd, 1) == 0);  
    mpz_invert(d, e, lambda);  
    mpz_clears(gcd, p_1, q_1, mul, mod, lambda, NULL);  
}
```

RSA (cont'd)

```
void encrypt(mpz_t ciphertext, const mpz_t message,
             const mpz_t e, const mpz_t n) {
    mpz_powm(ciphertext, message, e, n);
}

void decrypt(mpz_t message, const mpz_t ciphertext,
             const mpz_t d, const mpz_t n) {
    mpz_powm(message, ciphertext, d, n);
}

int main() {
    mpz_t msg, n, d, e;
    mpz_init_set_ui(msg, 123);    mpz_init_set_ui(n, 323);
    mpz_init_set_ui(e, 5);       mpz_init_set_ui(d, 29);
    enc_dec(msg, n, e, d);
    mpz_clears(n, d, e, msg, NULL);
    return 0;
}
```

RSA (cont'd)

```
void enc_dec(const mpz_t message,  
             const mpz_t n, const mpz_t e, const mpz_t d) {  
    mpz_t cipher, recovered;  
    mpz_inits(cipher, recovered, NULL);  
    encrypt(cipher, message, e, n);  
    decrypt(recovered, cipher, d, n);  
    assert(mpz_cmp(message, recovered) == 0);  
    printf("Original message: %s\n", mpz_get_str(NULL, 0, message));  
    printf("Ciphertext: %s\n", mpz_get_str(NULL, 0, cipher));  
    printf("Decrypted message: %s\n", mpz_get_str(NULL, 0, recovered));  
    mpz_clears(cipher, recovered, NULL);  
}
```

gmpy2 examples (1/4)

- ✧ <https://gmpy2.readthedocs.io/en/latest/mpz.html>
- ✧

```
from gmpy2 import mpz, powmod, invert, num_digits
from gmpy2 import random_state, mpz_random, divm
from gmpy2 import next_prime, is_prime, add, sub, mul,
from gmpy2 import f_mod, c_mod, gcd, gcdext
```
- ✧

```
x = mpz(12345432123454321) # ctor from int (python's large int)
y = mpz('543212345678901') # ctor from string
print(f'x={x}({x.type}) y={y}')
# x=12345432123454321 (<class 'mpz'>) y=543212345678901
```
- ✧ Mixed integer comparison:

```
if x==12345:
    print('x==12345') # x==12345
else:
    print('x!=12345')
```

```
if x>y:
    print('x>y')
else:
    print('x<=y') # x<=y
```

gmpy2 examples (2/4)

- ✧ `p = 123457`
`print(f'Is {p} a prime? {is_prime(p)}') # True`
- ✧ `plen = num_digits(p,2)`
`print(f'length of {p} is {plen} bits')`
- ✧ `random_state = random_state()`
`r = mpz_random(random_state,100000) # 0..99999`
`print(f'r={r}')` # 98411
`p2 = next_prime(r) # next prime > r    98419`
- ✧ `z = 5 * x + add(mul(mpz(6), y), -23) # z=5*x+6*y-23`
`print(f'z={z}')` # 387628
`print(f'{z}%{p}={z%p}')` # 17257
`print(f'mod({z},{p})={mod(z,p)}')` # 17257
`print(f'f_mod({z},{p})={f_mod(z,p)}')` # 17257
`print(f'c_mod({z},{p})={c_mod(z,p)}')` # -106200

gmpy2 examples (3/4)

- ✧ `a = 234126*97`
`b = mpz(2314512341234)*97`
`print(f'powmod({a},{b},{p})={powmod(a,b,p)}') # 860688`
`print(f'gcd({a},{b})={gcd(a,b)}') # 194`
`(g, s, t) = gcdext(a,b)`
`print(f'gcdext({a},{b})=({g}, {s}, {t})') # (194, 566058347467, -57260)`
`print(f'Verification:{a}*{s}+{b}*{t}={g}')`
`# Verification:`
$$22710222*566058347467+224507697099698*-57260=194$$
- ✧ `ainverse1 = powmod(a,-1,p)`
`print(f'powmod({a},-1,{p})={ainverse1}')`
`print(f'Verification: {a}*{ainverse1}%{p}={a*ainverse1%p}')`
- ✧ `ainverse2 = invert(a,p)`
`print(f'invert({a},{p})={ainverse2}')`

gmpy2 examples (4/4)

✧ `x = divm(a, b, p)` $\# b * x = a \pmod{p}$

```
print(f'divm({a}, {b}, {p})={x}') # 30080
```

```
print(f'Verification: {b} * {x} % {p} = {b*x%p}')
```

```
print(f'      {a} % {p} = {a%p}')
```

✧ `binverse = invert(b,p)`

```
x2 = binverse * a % p
```

```
print(f'2nd Verification: binverse * a % p = {x2}') # 30080
```

Pohlig-Hellman Discrete Log

```
❖ from gmpy2 import mpz, powmod, mod, invert
❖ p=65537
  beta10=mpz(2)
  beta11=mpz(2)
  beta=mpz(2)
  alpha=mpz(3)
  ❖ print(powmod(alpha,16384,p))
    print(invert(256,p))

❖ beta12=beta11*powmod(alpha,-2048,p)
  print(f'beta12={beta12}')
  print(powmod(beta12,8,p))

❖ beta13=mod(beta12*powmod(alpha,-4096,p),p)
  print(f'beta13={beta13}')

❖ beta14=beta13
  beta15=mod(beta14*powmod(alpha,-16384,p),p)
  print(f'beta15={beta15}')
```