공개키 암호의 구현

Part 2.Ep 3: GMP 라이브러리
YouTube: https://youtu.be/iylH0te0Ujw
Git: https://github.com/minpie/CryptoCraftLab-minpie_public

한성대학교 HANSUNG UNIVERSITY

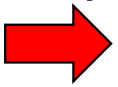
CryptoCraft LAB

발표 계획 목록

GMP 라이브러리

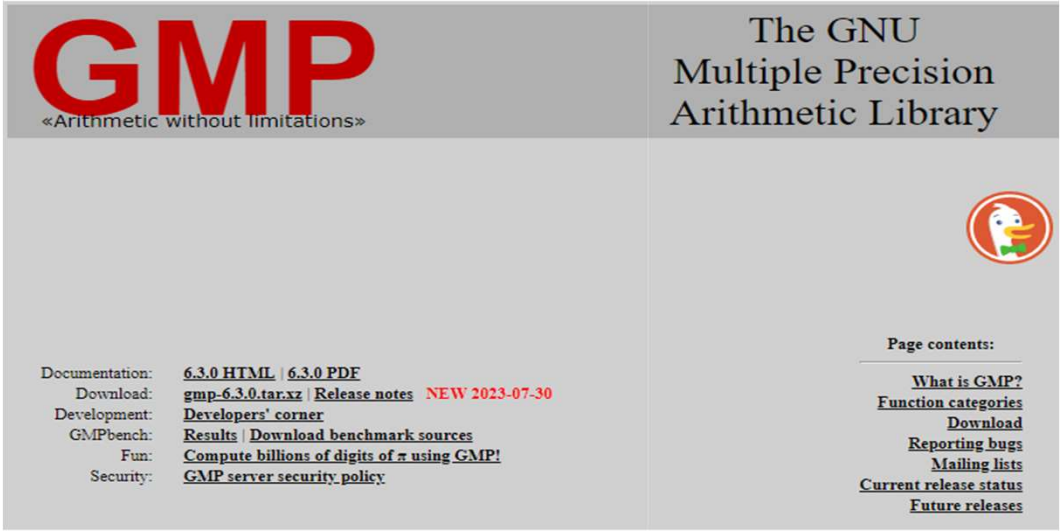
발표 계획: 24.07.19ver

- Part 1. 대칭키 암호 단일블록 C언어 구현
  - Ep1. AES
  - Ep2. DES
- Part 2. 64비트 이상 키 길이의 공개키 암호 C언어 구현

Today 

  - Ep3. GMP 라이브러리
  - Ep4. RSA 구현
  - Ep5. Rabin 구현
  - Ep6. Elgamal 구현
  - Ep7. ECDSA 구현
- Part 3. AES-운영모드 with 병렬컴퓨팅
  - Ep8. OpenMPI 라이브러리
  - Ep9. OpenMPI-AES
  - Ep10. CUDA C
  - Ep11. CUDA-AES

GMP 라이브러리 - 개요



- C언어에서 사용할 수 있는 임의 정밀도 수 계산 라이브러리
- 1991년 최초 발표
- https://gmplib.org/

GMP 라이브러리 – 공부의 필요성

```c
#include <stdio.h>
int main(void){
    printf("sizeof(unsigned long long int) = %ldByte\n", sizeof(unsigned long long int));
    printf("sizeof(double) = %ldByte\n", sizeof(double));

    return 0;
}
```

```
sizeof(unsigned long long int) = 8Byte
sizeof(double) = 8Byte
```

프로세서	12th Gen Intel(R) Core(TM) i7-12700H 2.30 GHz
설치된 RAM	32.0GB(31.7GB 사용 가능)
시스템 종류	64비트 운영 체제, x64 기반 프로세서

- C에서는 기본 정수 자료형의 최대 크기는 8바이트
- 이는 0 ~ $(2^{64}-1)$, 약 20자리의 수.

정수 상수에 대한 제한

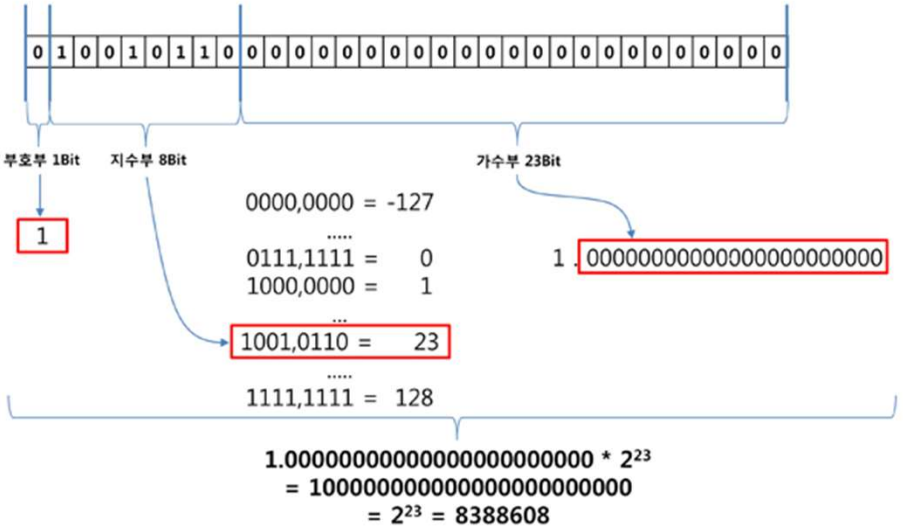
테이블 확장

상수	의미	값
LLONG_MAX	`long long` 형식 변수의 최대값입니다.	9,223,372,036,854,775,807
ULLONG_MAX	`unsigned long long` 형식 변수의 최대값입니다.	18,446,744,073,709,551,615 (0xffffffffffffffff)

GMP 라이브러리 – 공부의 필요성

부동소수점 표현 한계

부동소수점의 구조에서 알 수 있듯이, float이나 double의 경우 가수부의 크기는 일정하다. 따라서 지수가 충분히 클 경우에는 소수점 이하를 표현할 수 없게 된다.
무슨 의미인지 살펴보자!



- 부동 소수점 자료형은 더 큰 범위를 표현할 수 있으나, 표현 정확도가 일정 크기부터는 점점 낮아짐

<그림 4>를 살펴보자! 만일 지수부의 지수가 23을 나타낸다고 가정해보자! 가수부가 어떤 비트로 채워져 있건 상관없이 소수점이 오른쪽으로 23자리 이동하게 되므로 더 이상 소수점 이하 부분이 남아있지 않게 된다. 이것이 무슨 의미인가 하면 지수부의 지수가 23 이상일 경우에는 더 이상 float은 소수점 이하를 표현할 수 없다는 의미이다. 즉, 정수만을 표현하게 된다. 실제로 그림에서 가수부가 모두 0일 경우를 계산할 경우 float이 나타내는 값은 10진수로 8,388,608이 된다. 즉, float으로 8,388,608이상을 나타낼 때는 더 이상 소수점 이하를 표현할 수 없다는 것이다.

GMP 라이브러리 – 공부의 필요성

- 2진수의 실수 표현
  - 부동 소수점 형식의 표현

Positive range: $1.17549435 \times 10^{-38} \cdots 3.40282347 \times 10^{+38}$

그림 1-17 부동소수점 표현 형식

Positive range: $2.2250738585072014 \times 10^{-308} \cdots 1.7976931348623157 \times 10^{+308}$

서울여자대학교 SEOUL WOMEN'S UNIVERSITY

- 또한, double로도 1024비트 정도 이상의 크기의 수를 다루지 못함.

GMP 라이브러리 – 공부의 필요성

- 따라서 큰 수를 정확하게 다루기 위해서는 별도의 라이브러리 혹은 알고리즘이 필요
- 단순히 큰 수를 다룰 수 있다는 점 외에도 다양한 계산함수를 사용할 수 있음

Chapter 5: Integer Functions 39

`void mpz_lcm` (*mpz_t* **rop**, *const mpz_t* **op1**, *const mpz_t* **op2**) [Function]
`void mpz_lcm_ui` (*mpz_t* **rop**, *const mpz_t* **op1**, *unsigned long* **op2**) [Function]
Set *rop* to the least common multiple of *op1* and *op2*. *rop* is always positive, irrespective of the signs of *op1* and *op2*. *rop* will be zero if either *op1* or *op2* is zero.

`int mpz_invert` (*mpz_t* **rop**, *const mpz_t* **op1**, *const mpz_t* **op2**) [Function]
Compute the inverse of *op1* modulo *op2* and put the result in *rop*. If the inverse exists, the return value is non-zero and *rop* will satisfy $0 \leq rop < |op2|$ (with $rop = 0$ possible only when $|op2| = 1$, i.e., in the somewhat degenerate zero ring). If an inverse doesn't exist the return value is zero and *rop* is undefined. The behaviour of this function is undefined when *op2* is zero.

`int mpz_jacobi` (*const mpz_t* **a**, *const mpz_t* **b**) [Function]
Calculate the Jacobi symbol $(\frac{a}{b})$. This is defined only for b odd.

GMP 라이브러리 – 설치 방법

The library GMP provides the extended arithmetic for the number types `integer` and `rational`.

The following files should be available:

- `gmp.h`: the header files for the declarations exported from the library,
- `libgmp.[a,so,dyl]`: the library compiled either statically or dynamically.

1. Source code installation

The source code is available from the url http://gmplib.org/. The installation instructions are

```
./configure && make && make install
```

2. RPM installation

For a RPM installation with YUM, type

```
sudo yum install gmp
```

3. Debian installation

For a DEBIAN installation with APT-GET, type

```
sudo apt-get install libgmp3-dev
```

- 단순히 apt를 이용하거나 직접 컴파일을 하여 설치하는 방법이 존재.

Download the latest release of GMP

GMP 6.3.0	lz, 2086209 bytes	xz, 2094196 bytes	zstd, 2176751 bytes
Main site, gmplib.org, via https	gmp-6.3.0.tar.lz	gmp-6.3.0.tar.xz	gmp-6.3.0.tar.zst
USA, ftp.gnu.org, via https	gmp-6.3.0.tar.lz	gmp-6.3.0.tar.xz	gmp-6.3.0.tar.zst

To try to verify that the file you have downloaded has not been tampered with, you can chec
repackaging of gmp-5.1.0 as gmp-5.1.0a.tar.* the following key is used to sign GMP releases:

```
Key ID: 0x28C67298
Key type: 2560 bit RSA
Fingerprint: 343C 2FF0 FBEE 5EC2 EDBE F399 F359 9FF8 28C6 7298
```

Instead of using a release, you may also get the latest code from the **GMP repositories**. This wil

GMP 라이브러리 – 컴파일 방법

```
(py39gpu) watermark@watermark:/mnt/e/vscode/C/RSAcode$ gcc -o main_large_n_diy main_large_n_diy.c -lgmp
(py39gpu) watermark@watermark:/mnt/e/vscode/C/RSAcode$ ./main_large_n_diy
Done: Get n, phi_n
Done: Get e
Done: Get d
Public Key (n, e)=(115935041739676149688925098646158875237714573754541447754855261376147885408326350817276878815968325168468849300625485764111250162414552339182927162507656772727460097082714127730434960500556347274566628060099924037102991424472292215772798531727033839381334692684137327622000966676671831831088373420823444370953,
35535)
Private Key (d)=(58008302860037763936093661289677917594669062089650962180422866111380593852822358731706286910030021710859044338402170729869087600611530620252495988444804756824096624708148581713046324064407770483313401085094738529564507193677406119732655742423721761767462077637164207600337085333288532144708859551366702948 31)
Encryption: P=190708182608182600261904 1819, C=475309123646226827206365550610545180942371796070491716523239243054452960613199328566617843418359114151197411252005682979794571736036101278218847 8478927415660904800235071907152771859149751884658886321011483541033616578984679683867637337657774656250792805211481418440481418443081277305900469287424855916646210865669282742485591664621086569, P=190708182608182600261904 1819
(py39gpu) watermark@watermark:/mnt/e/vscode/C/RSAcode$
```

- gcc 컴파일 인자로는 –lgmp 사용(소문자 L)

GMP 라이브러리 – 사용 방법

```
mpz_t data_integer;          // 정수
mpq_t data_quotient;         // 유리수
mpf_t data_fp;               // 부동소수점 가수, short 크기
mp_exp_t data_exp;           // 부동소수점 지수, long 크기
mp_limb_t data_limb;         // 단일기계어에 맞는 다중 정밀도 숫자의 일부?, 32/64비트 크기
mp_size_t data_limb_size;    // Counts of mp_limb_t, int/long/long long 크기
mp_bitcnt_t data_bit_size;   // 다중 정밀도 숫자의 비트 수, 부호없는 long/부호없는 long long 크기
gmp_randstate_t data_rstate; // 알고리즘 선택/현재 상태 데이터?
```

- 수를 표현할때 Gmp만의 자료형을 선언하여 사용.

GMP 라이브러리 – 사용 방법

```
5   {
6       mpz_t n, phi_n;
7       mpz_t i;
8       mpz_t tmp1, tmp2;
9       mpz_inits(n, phi_n, NULL);
10      mpz_inits(i, NULL);
11      mpz_inits(tmp1, tmp2, NULL);
12
13      // get n , phi(n)
14      mpz_mul(n, p, q);           // n = p = q;
15      mpz_sub_ui(tmp1, p, 1);     // tmp1 = p - 1
16      mpz_sub_ui(tmp2, q, 1);     // tmp2 = q - 1
17      mpz_mul(phi_n, tmp1, tmp2); // phi_n = tmp1 * tmp2 = (p - 1) * (q - 1)
18      printf("Done: Get n, phi_n\n");
```

```
47      // end:
48      mpz_clears(n, phi_n, NULL);
49      mpz_clears(i, NULL);
50      mpz_clears(tmp1, tmp2, NULL);
51  }
```

- Gmp 자료형을 사용 전과 사용 종료 후 각각 mpz_init()와 mpz_clear()를 사용하여야 함.

GMP 라이브러리 – 사용 방법

```
77    mpz_set_str(plain, "190708182608182600261904181 9", 10);
78
79    // get n, e, d
80    mpz_mul(n, p, q); // n = p = q;
81    KeyGeneration(e, d, p, q);
82
83    gmp_printf("Public Key (n, e)=(%Zd, %Zd)\n", n, e);
84    gmp_printf("Private Key (d)=(%Zd)\n", d);
85
```

```
mpz_set_si(p, 397);
mpz_set_si(q, 401);
mpz_set_si(plain, 1314);
```

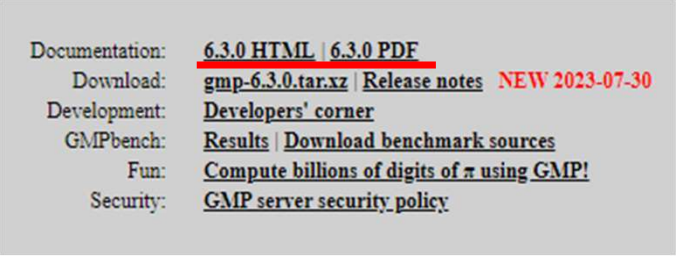
- Gmp 자료형에 값을 대입하기 위한 다양한 함수가 존재.
- Gmp 자료형의 값을 출력하기 위해서는 gmp_printf() 등을 %Zd 등과 같은 형식 지정자와 함께 사용.

GMP 라이브러리 – 사용 방법

```
53  void Encryption(mpz_t cipher, mpz_t plain, mpz_t n, mpz_t e)
54  {
55      mpz_powm(cipher, plain, e, n);
56  }
```

- Gmp 자료형을 직접 return 하거나 포인터와 함께 사용하는 것은 권장되지않음(그럴 필요가 없음)

GMP 라이브러리 – 사용 방법



- 이 외에도 많은 내용과 함수들은 전부 gmp 사이트에서 제공하는 온라인 페이지/pdf 에서 확인이 가능함.

Q & A