공개키 암호의 구현

Elgamal 구현
YouTube: https://youtu.be/xBHhmdKgrFo
Git: https://github.com/minpie/CryptoCraftLab-minpie_public

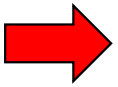
한성대학교 HANSUNG UNIVERSITY

CryptoCraft LAB

발표 계획 목록

Elgamal C언어 구현

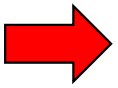
발표 계획: 24.07.19ver

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

발표 계획: 24.11.09ver

- 대칭키 암호 단일블록 C언어 구현
  - 1. AES
  - 2. DES
- 64비트 이상 키 길이의 공개키 암호 C언어 구현
  - 3. GMP 라이브러리
  - 4. RSA 구현
  - 5. Rabin 구현
  - 6. Elgamal 구현
  - 7. ECDSA 구현
- AES-운영모드 with 병렬컴퓨팅
  - 8. OpenMPI 라이브러리
  - 9. OpenMPI-AES
  - 10. CUDA C
  - 11. CUDA-AES

Today 

Elgamal C언어 구현 - 개요


IEEE TRANSACTIONS ON INFORMATION THEORY, VOL. IT-31, NO. 4, JULY 1985

A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms

TAHER ELGAMAL, MEMBER, IEEE

- 기본적인 수준에서 공개키 암호로서의 Elgamal을 구현
- 기반문제: 이산로그 문제

Elgamal C언어 구현 – 전체 흐름

```c
int main(void)
{
    mpz_t plain, cipher1, cipher2, e1, e2, p, plain2, d;
    mpz_inits(plain, cipher1, cipher2, e1, e2, p, plain2, d, NULL);

    /*
    // test:
    mpz_set_ui(plain, 7);
    mpz_set_ui(p, 11);
    */


    // test:
    mpz_set_str(p, "115348992725616762449253137170143317404900945326098349598143469
    mpz_set_ui(plain, 3200);

    // Encrypt & Decrypt:
    KeyGeneration(e1, e2, d, p);
    Encrypt(cipher1, cipher2, plain, e1, e2, p);
    Decrypt(plain2, cipher1, cipher2, d, p);


    // print result:
    printf("KeyGeneration:\n");
    printf("Public Key:\n");
    gmp_printf("e1 = %Zd, e2 = %Zd, p = %Zd\n", e1, e2, p);
    printf("\n\n");
    printf("Private Key:\n");
    gmp_printf("d = %Zd\n", d);

    printf("\n\n");
    printf("Encrypt:\n");
    gmp_printf("Plain=%Zd, Cipher1=%Zd, Cipher2=%Zd\n", plain, cipher1, cipher2);

    printf("\n\n");
    printf("Decrypt:\n");
    gmp_printf("Plain=%Zd, Cipher1=%Zd, Cipher2=%Zd\n", plain2, cipher1, cipher2);

    mpz_clears(plain, cipher1, cipher2, e1, e2, p, plain2, d, NULL);
    return 0;
}
```

Elgamal C언어 구현 – KeyGeneration()

```
90  void KeyGeneration(mpz_t e1, mpz_t e2, mpz_t d, mpz_t p)
91  {
92      mpz_t i, i_end, phi_p, tmp1;
93      mpz_inits(i, i_end, phi_p, tmp1, NULL);
94      // Get d:
95      mpz_set_ui(i, 1);         // i = 1
96      mpz_sub_ui(i_end, p, 2); // i_end = p - 2
97      /*
98      while (mpz_cmp(i_end, i) >= 0)
99      {
100         mpz_set(d, i); // d = i = d in Zp*
101
102         mpz_add_ui(i, i, 1); // i++
103     }
104     */
105     // mpz_set_ui(d, 3); // test
106     mpz_set_ui(d, 1007); // test
107
108     // Get e1:
109     mpz_sub_ui(phi_p, p, 1); // phi_p = p - 1
110     mpz_set_ui(i, 1);         // i = 1
111     mpz_sub_ui(i_end, p, 1); // i_end = p - 1
112
113     /*
114     while (mpz_cmp(i_end, i) >= 0)
115     {
116         GetElementOrd(tmp1, i, p);
117         if (!mpz_cmp(tmp1, phi_p))
118         {
119             // tmp1 == phi_p:
120             mpz_set(e1, i);
121         }
122
123         mpz_add_ui(i, i, 1); // i++
124     }
125     */
126     // mpz_set_ui(e1, 2); // test
127     mpz_set_ui(e1, 2); // test
128     // Get e2:
129     FastModuloExponentiation(e2, e1, d, p);
130     // end:
131     mpz_clears(i, i_end, phi_p, tmp1, NULL);
132 }
```

- 키생성 연산

Elgamal C언어 구현 – FastModuloExponentation()

```
4   // 고속 모듈러 지수연산
5   void FastModuloExponentiation(mpz_t result, mpz_t a, mpz_t x, mpz_t n)
6   {
7       mp_bitcnt_t bits_x, i;
8       mpz_t tmp_a, y;
9       mpz_inits(tmp_a, y, NULL);
10      mpz_set(tmp_a, a);
11      mpz_set_ui(y, 1); // y = 1
12      i = 0;
13      bits_x = mpz_sizeinbase(x, 2);
14
15      while (bits_x > i)
16      {
17          if (mpz_tstbit(x, i))
18          {
19              // x의 i번째 비트가 1이면:
20              mpz_mul(y, tmp_a, y); // y = tmp_a * y
21              mpz_mod(y, y, n);     // y = y mod n
22          }
23          mpz_mul(tmp_a, tmp_a, tmp_a); // tmp_a = tmp_a * tmp_a
24          mpz_mod(tmp_a, tmp_a, n);     // tmp_a = tmp_a mod n
25          i = (mp_bitcnt_t)(i + 1);     // i++
26      }
27      mpz_set(result, y);
28      mpz_clears(tmp_a, y, NULL);
29  }
```

- 모듈로 거듭제곱 함수
- 제곱-곱 방법을 이용

Elgamal C언어 구현 – Encrypt()

```
141  void Encrypt(mpz_t cipher1, mpz_t cipher2, mpz_t plain, mpz_t e1, mpz_t e2, mpz_t p)
142  {
143      mpz_t r, tmp1, tmp2;
144      gmp_randstate_t randstate;
145      mpz_inits(r, tmp1, tmp2, NULL);
146      gmp_randinit_default(randstate);
147
148      // Get r:
149      mpz_sub_ui(tmp2, p, 1);           // tmp2 = p - 1
150      mpz_urandomm(r, randstate, tmp2); // r = 0 ~ (tmp2 - 1) = 0 ~ (p - 2)
151      mpz_add_ui(r, r, 1);              // r = r + 1 = 1 ~ (p - 1)
152
153      //mpz_set_ui(r, 4); // test
154      mpz_set_ui(r, 545131); // test
155
156      // Get cipher1:
157      FastModuloExponentiation(cipher1, e1, r, p);
158
159      // Get cipher2:
160      FastModuloExponentiation(tmp1, e2, r, p);
161      mpz_mul(cipher2, plain, tmp1);
162      mpz_mod(cipher2, cipher2, p);
163
164      mpz_clears(r, tmp1, tmp2, NULL);
165  }
```

- 암호화 연산

Elgamal C언어 구현 – Decrypt()

```
167 void Decrypt(mpz_t plain, mpz_t cipher1, mpz_t cipher2, mpz_t d, mpz_t p)
168 {
169     mpz_t tmp1, cipher1_d_1;
170     mpz_inits(tmp1, cipher1_d_1, NULL);
171
172     FastModuloExponentiation(tmp1, cipher1, d, p);
173     GetModularMultiplicativeInverse(cipher1_d_1, p, tmp1);
174     mpz_mul(tmp1, cipher2, cipher1_d_1);
175     mpz_mod(plain, tmp1, p);
176
177     mpz_clears(tmp1, cipher1_d_1, NULL);
178 }
```

- 복호화 연산

Elgamal C언어 구현 – GetModularMultiplicativeInverse()

```
52  // 모듈러 곱셈 역 구하기
53  void GetModularMultiplicativeInverse(mpz_t a_1, mpz_t n, mpz_t a)
54  {
55      // 확장 유클리드 알고리즘 사용
56      mpz_t q, r1, r2, r, t, t1, t2, tmp1, tmp2;
57      mpz_inits(q, r1, r2, r, t, t1, t2, tmp1, tmp2, NULL);
58      mpz_set(r1, n);    // r1 = n;
59      mpz_set(r2, a);    // r2 = a
60      mpz_set_si(t1, 0); // t1 = 0
61      mpz_set_si(t2, 1); // t2 = 1
62
63      while (mpz_cmp_si(r2, 0))
64      {
65          mpz_fdiv_q(q, r1, r2); // q = r1 / r2
66          mpz_mul(tmp1, q, r2);  // tmp1 = q * r2
67          mpz_sub(r, r1, tmp1);  // r = r1 - tmp1 = r1 - (q * r2)
68          mpz_set(r1, r2);       // r1 = r2
69          mpz_set(r2, r);        // r2 = r
70
71          mpz_mul(tmp2, q, t2); // tmp2 = q * t2
72          mpz_sub(t, t1, tmp2); // t = t1 - tmp2 = t1 - (q * t2)
73          mpz_set(t1, t2);      // t1 = t2
74          mpz_set(t2, t);       // t2 = t
75      }
76      mpz_set(a_1, t1);
77      if (mpz_sgn(a_1) < 0)
78      {
79          mpz_add(a_1, n, a_1);
80      }
81      mpz_clears(q, r1, r2, r, t, t1, t2, tmp1, tmp2, NULL);
82  }
```

- 모듈로 곱 역원을 구하는 함수
- 확장 유클리드 알고리즘 사용

Elgamal C언어 구현 – 실행 결과

```
watermark@watermarkserver:/storage/drive1/pt1/codes/C/ElGamalCryptosystem$ ./main
KeyGeneration:
Public Key:
e1 = 2, e2 = 358848275692829563014094368505161589436729771803011287416646450973623548018370807608769
674858681528750303412415496115960368675168682226038935309399681396 2, p = 115348992725616762449253137
170143317404900945326098349598143469219056898698622645932129754737871895144368891765264730936159299 9
372806116596434735344000857 7


Private Key:
d = 1007


Encrypt:
Plain=3200, Cipher1=887297069383528471022570471492275663120260067256562125018188351429417223599712 68
111410536366170517305158153318916540097373635508029573678854690606191528 81, Cipher2=4543423782546875
749145490117569412850013029593221655431059251673935287629973791152332793708204374930299269302444285 0
420916630746865610266336422373030217 55


Decrypt:
Plain=3200, Cipher1=887297069383528471022570471492275663120260067256562125018188351429417223599712 68
111410536366170517305158153318916540097373635508029573678854690606191528 81, Cipher2=4543423782546875
749145490117569412850013029593221655431059251673935287629973791152332793708204374930299269302444285 0
420916630746865610266336422373030217 55
watermark@watermarkserver:/storage/drive1/pt1/codes/C/ElGamalCryptosystem$
```

- 처음 평문을 암호화 하고 다시 복호화 해서 원래 평문이 나옴으로서 알고리즘이 잘 동작한 것을 확인

Elgamal C언어 구현 – 참고문헌

- https://en.wikipedia.org/wiki/ElGamal_encryption
- https://caislab.kaist.ac.kr/lecture/2010/spring/cs548/basic/B02.pdf
- https://cacr.uwaterloo.ca/hac/about/chap8.pdf

Q & A