정보암호화 결과보고서

타원곡선암호(Elliptic Curve Cryptography; ECC) 알고리즘 구현

5조

2018.06.07.

전공	학번	이름
미디어기술콘텐츠학과	201620031	강주연
정보통신전자공학부	201620693	서지은
정보통신전자공학부	201121469	전호진
정보통신전자공학부	201621570	한시연

목 차

요약

타원곡선암호(Elliptic Curve Cryptography, ECC)는 타원곡선 이론에 기반을 둔 공개키 암호 방식이며, 1985년 Koblitz와 Miller에 의하여 제안되었다. 타원곡선을 이용한 암호 방식의 대표적 장점은 기존 공개키 암호 시스템, 이를테면 RSA, ElGamal등과 같은 방식에 비해 비교적 짧은 키를 사용하면서도 그와 비슷한 수준의 안정성을 제공한다는 것이다.

본 보고서에서는 강의시간에 다루지 않았던 공개키 알고리즘인 이 타원곡선암호화 기법을 직접 MATLAB 프로그램을 통해 구현해봄으로써 실제로 타원곡선암호가 어떤 순서에 의해 실현되는지에 대한 연구 과정과 결과를 보여주고자 한다.

1. 소개

본 연구의 첫 번째 목적은 타원곡선암호의 의미와 차별점에 대해서 파악해보는 것이다. 다음으로 타원곡선암호를 실제 MATLAB 상에서 구현하기 위한 접근 방식을 탐구해보고, 이에 연계하여 단계적으로 따르는 문제점과 해결방안들에 대하여 고찰해보고자 한다. 이 과정에서 실제로 타원곡선암호 방식이 어떤 순서도를 갖는지, 어떤 코드 매핑을 따라야하는지 확인할 수 있을 것이다.

2. 타원곡선암호

타원곡선암호기술(Elliptic Curve Cryptography, ECC)는 타원곡선 이론에 기반을 둔 공개키 암호 방식으로, 1985년 Koblitz와 Miller가 RSA암호방식의 대안으로 제안하였다. 기존 RSA암호방식에는 암호키의 길이가 길어지면 보안성은 강화 되지만, 암호연산 속도가 느려지는 문제점이 있는데, 이를 보완하기 위해 ECC 방식을 사용하는 추세이다. 이는 ECC를 사용하면 RSA보다 적은 bit수의 암호키로 RSA와 동일한 수준의 암호성능을 낼 수 있는 동시에 암호키의 bit수가 적어서 암호 연산 속도가 상대적으로 빨라지기 때문이다. 예를 들어 3072-bit RSA방식과 256-bit ECC 방식의 암호성능이 동일하다고 볼 수 있다.

2-1. 타원곡선

타원곡선은 방정식 $y^2 = x^3 + ax + b$ 로 정의되는 대수 곡선으로, 첨점이나 교차점 등의 특이점이 없다.

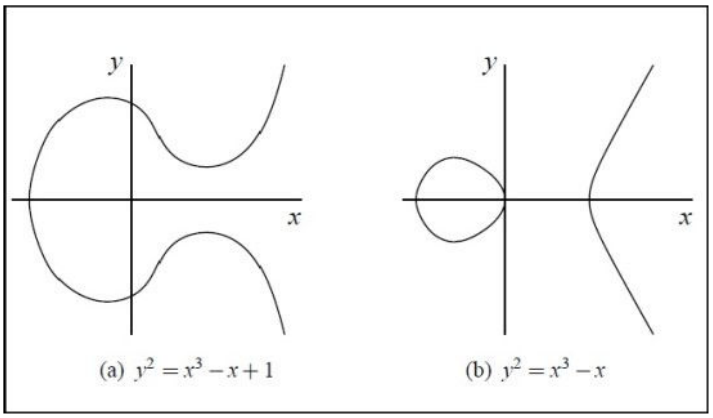
소수(Prime) p 를 사용하는 유한체 상의 타원곡선을 수식으로 표시하면 아래와 같고,

y^2 = x^3 + ax + b over Fp 또는
y^2(mod p) = x^3 + ax + b(mod p)

집합으로 표시하면 아래와 같다.(무한원점도 타원곡선에 포함된다고 정의하기 때문에, 무한원점을 표시하는 {O}도 포함되어 있다). x 값과 y 값이 정수이기 때문에 (x , y)값을 점으로 표시하면 곡선이 아니라 흩어져 있는 점들로 보인다.

$$E(F_P) = (x,y)|y^2 = x^3 + ax + b \cup O$$

a 와 b 의 값에 따라 곡선의 모양이 다르며, 대표적인 두 개만 나타내면 아래와 같다.



< 대표적인 타원곡선 예시 >

2-1-1. 타원곡선 특징

타원곡선은 x축을 기준으로 대칭적이다. y=0을 만족하는 방정식의 해를 구하면 하나의 실수와 두 개의 복소수(좌표에는 나타나지 않음)가 존재한다. 점 P(x1 , y1)과 점 Q(x2, y2)를 더하기 위해서 P와 Q를 잇는 선을 그으면 타원곡선 위의 다른 점 R과 교차한다. 만약 P=Q이면 점 P에 대한 접선을 그으면 된다. 계산한 점 R을 X축에 대칭을 시킨 다른 점 S가 P+Q로 정의된다.

2-1-2. 타원곡선 군에서의 덧셈의 특성

(1) 무한대 점은 O로 표기하며 타원곡선 위의 임의의 점 P에 대해서 P+O=P가 성립된다.
즉, 무한대 점은 덧셈상의 항등원이 된다.

(2) 타원곡선 위의 임의의 점 P에 대해 P+Q=O를 만족하는 점 Q를 Q=-P로 나타내며 뺄셈 P-Q는 P+(-Q)로 정의된다. 점 -P는 점 P의 X축의 대칭점이 되기 때문에 점 P의 좌표가 (x, y)이면 점 -P의 좌표는 (x, -y)이 된다.

(3) 타원곡선 위의 임의의 점 P, Q, R에 대해 P+(Q+R)=(P+Q)+R이 성립하고(결합법칙) P+Q=Q+P가 성립한다(교환법칙). 그러므로 타원곡선 군은 "가환군"이 된다.

2-2. 타원곡선암호의 현재와 미래 활용도

ECC방식이 적용되는 분야와 알고리즘은 다음과 같다.

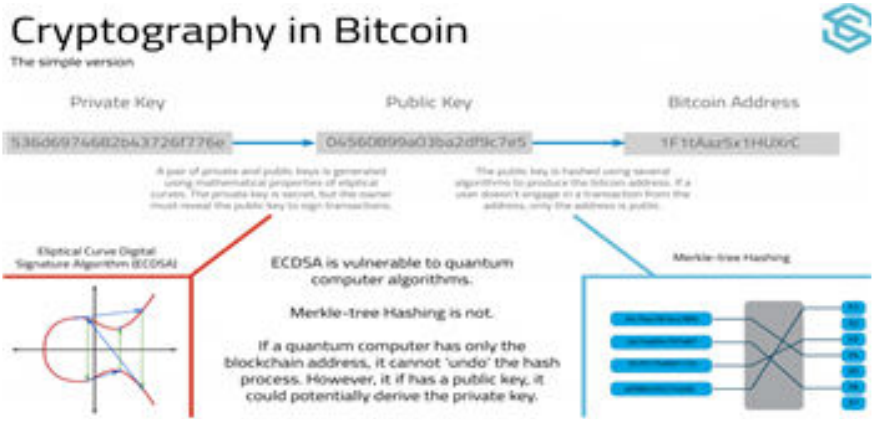
(1) 디지털서명 용도 : ECDSA

(2) 키교환 용도 : ECDH

(3) 난수 생성 용도: Dual-EC_DRBG

(4) Encryption & Decryption 용도

(5) Bitcoin



< 비트코인에 활용된 타원곡선암호 >

위 그림과 같이 타원곡선은 현재 비트코인의 암호화 기술에도 사용되고 있으며, 블루투스 무선 프로토콜과 같은 연산 및 정보처리 능력이 제한적인 모바일 환경에서 데이터 암호화 또는 키 관리 방식으로도 응용되고 있다. 국내 여러 보안업체들도 스마트폰 환경에서 타원곡선을 구현하는 보안제품을 개발을 추진하고 있다. IC카드는 메모리 용량의 한계로 연산 능력이 제한적인데, 미래에는 이처럼 작은 메모리가 장착된 IC카드 내의 정보를 암호화하기 위한 방식으로도 타원곡선 알고리즘을 구현할 수 있을 것이다.

3. 실험 결과

3-1. 공개키, 개인키 생성

키 생성은 공개키(비대칭키) 암호화 방식의 중요한 부분으로, 공개키는 메시지를 암호화하는데 사용되고, 개인키는 메시지를 복호화 하는데 사용된다. 이때 공개키로 암호화된 메시지는 개인키를 통해서만 복호화가 가능하다. 다음은 공개키와 개인키를 생성하는 방법이다.

개인키(d) : 1 ~ (n-1) 사이의 값

(n: 위수)

공개키(Q) : Q(x, y) = d * G(x0, y0)

(d: 개인키, G: 타원곡선 위에 점 중 하나로 base point)

다음은 MATLAB을 이용하여 우리가 사용할 타원곡선의 파라미터(p_소수, a,b_타원곡선 정의 계수, G_base point, n_order of G)를 지정하고 공개키와 개인키를 생성하는 코드(public_parameter.m)이다.

```
================================================================================
clc; clear;

%%%% 파라미터 지정
%%%% p_소수 // a,b_타원곡선 정의 계수 // G_base point // n_order of G
%%%% 발표용 p = 599 / a = 2 / b = 0 // p는 p=(4*i)-1을 만족해야 한다
p = 599; a = 2; b = 0;
find_dot(p,a,b);
load('parameter.mat');
[G n] = generate_G(dot,p,a);

%%%% 개인키 d
d = 59;
%%%% 공개키 Q
Q = multiple_dot(d,G,p,a);

save('public_parameter.mat','a','b','p','Q','G');
================================================================================
```

3-2. 암호화 과정

m이 보내는 메시지일 때 메시지를 타원 곡선 위의 점 M으로 mapping할 수 있다.(3-4-6 참고) k는 랜덤으로 선택된 1과 n-1 사이의 값이다. 다음은 메시지를 mapping한 점 M을 C1, C2로 암호화하는 방법이다.

$$C1 = k*P \qquad C2 = M + k*Q$$

다음은 위 방법에 따라 공개된 정보(p,a,b,G,Q)를 이용하여 평문(ecc_plain.txt)을 암호화하는 코드(Encrypt_ECC.m)이다. 평문은 암호화 과정을 거쳐 암호문(ecc_cipher.txt)에 저장된다.

```
=========================================================================
%%% 공개된 정보(p,a,b,G,Q)를 읽어옴
load('public_parameter.mat');

%%% 암호화 할 텍스트 파일을 읽어와서 문자열을 plain_text에 저장
fpt = fopen('ecc_plain.txt','r');
plain_text = fread(fpt);
fclose(fpt);

%%% plain_text의 각 문자를 ASCII로 인코딩한 후, 이를 곡선위의 점으로 mapping
%%% 문자 하나당 점 하나로 mapping됨
dots = mapping(plain_text, p);

%%%% encrypt dots
%%% [1,n-1]의 난수 k. n은 G와 p,a,b를 알면 구할 수 있음
k = 15;
C1 = multiple_dot(k,G,p,a);
kQ = multiple_dot(k,Q,p,a);
C2 = dots + kQ;

%%% C1과 C2를 16진수로 바꾸어 텍스트로 저장
fpt = fopen('ecc_cipher.txt','w');
fprintf(fpt,'%04x',C1);
fprintf(fpt,'%04x',C2');
fclose(fpt);

fprintf('Encryption is finished :D\n');
=========================================================================
```

3-3. 복호화 과정

다음은 암호문을 평문 M으로 복호화 하는 방법이다.

M = C2 - d*C1 (k*Q = d*P*k = C1*d)

이는 다음과 같이 증명 가능하다.

C2 - d * C1 = (M + k*Q) - d*(k*P) (C2 = M + k*Q , C1 = k*P)
= M + k*d*P - d*k*P (k*d*P = d*k*P는 지워짐)
= M (Original Message)

다음은 위 방법에 따라 암호문(ecc_cipher.txt)을 원래의 평문으로 복호화 하는 코드(Decrypt_ECC.m)이다. 복호화 된 코드(ecc_decrypt.txt)는 평문(ecc_plain.txt)과 같아진다.

```
================================================================================
%%% 암호화된 메시지를 읽어옴
fpt = fopen('ecc_cipher.txt','rt');
cipher = fread(fpt,'int8=>char');
fclose(fpt);

%%% 저장된 형식에 따라 C1과 C2를 읽어옴
%%% C1의 x좌표는 처음 4개문자(숫자) C1의 y좌표는 그 다음 4개문자(숫자)
c1 = [hex2dec(cipher(1:4)'), hex2dec(cipher(5:8)')];

%%% C2의 각 x,y좌표들은 4개문자(숫자)가 번갈아가면서 나옴
c2 = [0 0];
idx = 9;
while 1
    idx1 = idx+3;
    idx2 = idx+4;
    idx3 = idx+7;
    x = hex2dec(cipher(idx:idx1)');
    y = hex2dec(cipher(idx2:idx3)');
    c2 = [c2; x y];
    idx = idx+8;
    if idx3 == size(cipher,1)
        break;
    end
```

```
end
c2 = c2(2:end,:);

%%% c1, c2를 이용한 복호화
dc1 = multiple_dot(d,c1,p,a);
result = c2 - dc1;

%%% result는 c2에서 얻어낸 매핑된 점의 집합
%%% result(1)은 x좌표, result(2)는 y좌표
%%% y좌표가 p/2 보다 작으면 result(1)이 원문의 ASCII값
%%% y좌표가 p/2 보다 크면 p-result(1)이 원문의 ASCII값
for i=1:size(result,1)
    if result(i,2)>p/2
        origin_M(i) = p-result(i,1);
    else
        origin_M(i) = result(i,1);
    end
end

%%% 복호화한 텍스트를 저장
fpt = fopen('ecc_decrypt.txt','w');
fprintf(fpt,'%c',origin_M);
fclose(fpt);

fprintf('Decryption is finished :D\n');
================================================================================
```

3-4. 암/복호화 과정에서 사용한 함수

3-4-1. add_dot.m

P, Q, p, a를 받아 타원곡선 위의 점 P, Q를 타원곡선 $y^2 = x^3 + ax$ 상에서 더한다.

```
================================================================================
function result = add_dot(P, Q, p, a)

%%% infinity에 대한 처리
if P==[0,0]
    result = [Q(1), Q(2)];
    return
elseif Q ==[0,0]
    result = [P(1), P(2)];
    return
end

%%% infinity가 아닌 경우
x1 = P(1);
y1 = P(2);
x2 = Q(1);
y2 = Q(2);

if (x1==x2)&&(y1~=y2)
    result = [0, 0];
    return
end

if(x1==x2&&y1==y2)
      s=mod(mod(3*x1*x1+a,p)*Inv(p,mod(2*y1,p)),p);
else
      s=mod(mod(y2-y1,p)*Inv(p,mod(x2-x1,p)),p);
end

x3=mod(s*s-x1-x2,p);
y3=mod(s*(x1-x3)-y1,p);
result=[x3, y3];
end
================================================================
```

3-4-2. check_n.m

G, dot, p, a를 받아 타원 곡선 Base point(G)의 위수 n을 구한다. add_dot을 하는 과정에서 나오는 점과 타원곡선 위에 점 리스트(dot)를 비교하여 위수 n값을 구한다.

```
================================================================================
function n = check_n(G,dot,p,a)
n=1;
R = G;

while 1
    n = n+1;
    r = add_dot(G,R(end,:),p,a);


    if ismember(r,dot,'row')
        R = [R; r];
    else
        %fprintf("dot(" + r(1) + "," + r(2) + ") is not on curve")
        break;
    end

    if r == [0,0];
        break;
    elseif r(2)==0 && R(end-1,2)==0
        break;
    end
end
end
================================================================================
```

3-4-3. find_dot.m

p, a, b를 받아 정의된 타원곡선 $y^2 = x^3 + ax + b$상의 군의 원소를 찾는다.

```
=========================================================================
function find_dot(p,a,b)
%%% 기본 인자 (필요에 따라 변경)
%%%p = 17; a = 2; b = 1;

%%% x = [0, p-1] 에서 나오는 ( 우변 mod p )의 배열 Right
%%% y = [0, p-1] 에서 나오는 ( 좌변 mod p )의 배열 Left
for i=0:p-1
    Right(i+1) = mod( i^3 + (a*i) + b , p);
    Left(i+1) = mod(i^2, p);
end

%%% Right와 Left는 유지하고 (x,y좌표를 알아야 하므로!)
%%% Right와 Left의 중복되는 값 찾기 (unique와 intersect 이용)
R = unique(Right);
L = unique(Left);
inter = intersect(R,L);

%%% 중복되는 값의 인덱스 찾아서 좌표점 완성
dot = [0 0];
for i=1:size(inter,2)
   idx_x = find(Right == inter(i));
   idx_y = find(Left == inter(i));
   for j=1:size(idx_x,2)
       for k=1:size(idx_y,2)
           dot = [dot; idx_x(j)-1 idx_y(k)-1];
       end
   end
end

dot = dot(2:end, :);
save('parameter.mat','a','b','p','dot');
end
=========================================================================
```

3-4-4. generate_G.m

타원곡선 위에서 찾은 점 중 적당한 base point를 찾는다. base point의 위수는 타원곡선 위에 존재하는 전체 점의 개수의 1/4이상이어야 한다. 특히 위수가 클수록 좋기 때문에 타원곡선 위에 점 중 큰 위수를 가지는 점을 base point(G)로 하였다.

```
================================================================================
function [G, max_n] = generate_G(dot,p,a)
k = size(dot,1);
max_n = 0;
for i=1:2:k
    n = check_n(dot(i,:),dot,p,a);
    if n>max_n;
        max_n = n;
        G = dot(i,:);
        if max_n==k
            break;
        end
    end
end
end
================================================================================
```

3-4-5. Inv.m

음수가 아닌 정수 m, a를 받아 mod m 상에서 a의 역원을 구한다, add_dot.m 함수에서 사용된다.

```
 ================================================================
function result = Inv(m,a)

% memory assignment
r=zeros(1,50);
q=zeros(1,50);
u=zeros(1,50);
v=zeros(1,50);

% initialization
u(1)=0; u(2)=1;
v(1)=1; v(2)=0;
q(1)=0; q(2)=0;
r(1)=m; r(2)=a;
```

```
% iteration
i=2;
while r(i)~=0
    i=i+1;
    r(i)=mod(r(i-2),r(i-1));
    q(i)=floor(r(i-2)/r(i-1));
    u(i)=u(i-2)-u(i-1)*q(i);
    v(i)=v(i-2)-v(i-1)*q(i);
end

N=i-1;
if r(N)==1
   if u(N)<0 || u(N)>=m
      if u(N)<0
         u(N)=u(N)+m;
      else
         u(N)=u(N)-m;
      end
   end
else
   u(N)=0;
end

result=u(N);
end
```

==========

3-4-6. mapping.m

text와 p를 받아 text의 문자 m을 타원곡선 상의 한 점 M으로 mapping한다. 이때 mapping 과정은 Modified Koblitz Encoding Method for ECC(Kodali, Sarma)에서 제안된 방식을 이용하였다.

타원 곡선을 정의하는 a와 b에서 b=0이고 소수 p가 p=(4*i)-1을 만족하면 (1) y^2 mod p = r 일 때 Fp상의 다른 어떤 y도 y^2 mod p = (p-r)을 만족하지 않고 (2) f(x)=x^3 + ax + b에 대해 f(j) mod p = t 이면 f(p-j) mod p = p - t 임이 증명되어 있다. 즉, y^2 mod p = r인 y1, y2가 존재하면 y1과 y2를 제외한 다른 y는 y^2 mod p = p-r을 만족하지 않으며, f(x) = t mod p이면 f(p-x) = p - t mod p가 성립한다. 이 두가지 성질을 이용해 메시지를 곡선위의 점으로 mapping하고 곡선 위의 점으로 부터 메시지를 reverse mapping 할 수 있다. 방법은 다음과 같다.

(1) mapping 하려는 메시지 m의 i번째 글자를 정수로 표현한다. 이는 문자를 인코딩한 결과(ASCII 등)일 수도 있고, a=1, b=2,... 과 같이 송,수신자 간의 약속된 값일 수도 있다. 이 정수를 X라 하자.

(2-1) 타원 곡선 y^2 mod p = x^3 + ax + b mod p 위에 점 (X, Y1)과 점 (X, Y2)가 존재하면 M=(X, Y1)으로 mapping한다. (Y1<Y2)

(2-2) 만약 점 (X, Y1)과 점 (X, Y2)가 존재하지 않는다면 점 (p-X, Y1)과 점 (p-X, Y2)가 존재하므로 M=(p-X,Y2)으로 mapping한다. (Y1<Y2)

이렇게 mapping된 점은 y값에 따라 reverse mapping이 가능하다. mapping 과정에서 Y1<Y2이고 Y1+Y2=p이므로 Y1 < p/2 이고, Y2 > p/2 이다. 이를 통해 mapping된 점 (X', Y')에서 Y'<p/2이면 원래 문자가 정수로 표현된 값은 X'이고, Y'>p/2이면 원래 문자가 정수로 표현된 값은 p-X'임을 알 수 있다.

```
=====================================================================
function result = mapping(text,p)
%%%% 메시지 매핑
%%%% Modified Koblitz Encoding Method for ECC
%%%% 2013 ACEEE
%%%% Ravi Kishore Kodali and Prof. Narasimha Sarma NVS
%%%% National Institute of Technology, Warangal
%%%% Department of  E. and C. E., N.I.T., Warangal, India

%%%% 필요한 변수!
load('parameter.mat')
%%%% result 틀 마련
N = length(text);
```

```
result = [0,0];
%%%%% X,Y computation example
for i=0:p-1
    j=i+1;
    X(j) = mod(i^3+a*i+b,p);
    Y(j) = mod(i^2,p);
end

for i=1:N
    M = text(i); %% ASCII (0~128 중 문자 하나!), 97이면 'a'
    if ismember(X(M+1), Y)
        idx = find(Y==X(M+1));
        m = [M, idx(1)-1];
    else
        idx = find(Y==X(p-M+1));
        m = [p-M, idx(2)-1];
    end
    result = [result; m];
end
result = result(2:end,:);
end
```

==========

3-4-7. multiple_dot.m

times, dot, p, a를 받아 times*dot연산을 한다.

```
==========
function result = multiple_dot(times,dot,p,a)
k = times;
tmp_dot = dot;
if(k>1)
    for i=1:k-1
        tmp_dot = add_dot(dot,tmp_dot,p,a);
    end
end
result = tmp_dot;
end
```

==========

3-5. 실행 예시

먼저 우리가 사용할 타원곡선의 파라미터를 지정하고 공개키와 개인키를 생성하기 위해 public_parameter.m을 실행한다. 실행하면 parameter.mat과 public_parameter.mat이 생성된다. 그 후 우리가 원하는 ecc_plain.txt에 평문을 다음과 같이 작성하고,

Elliptic Curve Cryptography

Encrypt_ECC.m을 실행하면 평문은 암호화 과정을 거쳐 암호문 ecc_cipher.txt으로 저장된다.

00dc00f60438016d0292007002920070041402260040d0190029a0069041402260289 00fb0246004b043a0253029b001b040b01ec029c0090041801cb0246004b043a0253040b01ec040401d0040d0190029a0069040e020a0416024a040b01ec041c0264040d0190028e013b040401d0

암호문은 다시 복호화를 위해 Decrypt_ECC.m를 실행하면 하면 ecc_decrypt.txt에 다시 평문이 저장된 것을 볼 수 있다.

Elliptic Curve Cryptography

4. 고찰

4-1. MATLAB 프로그램의 한계

예비 보고서 작성시에는 190bit 기반 ECC를 구현하려고 계획했지만 구현하지 못했다. 이는 매트랩에서 제공하는 정수 크기가 제한되어 있기 때문인데, 매트랩에서는 자료형의 변환 없이 (10억-1)까지의 수를 정수로 연산할 수 있고 그 이상의 수에 대해서는 배정밀도의 부동소수점 방식을 사용하므로 정밀도가 떨어지게 된다. 이러한 정밀도의 감소는 2^190에 대해서는 2^190과 2^190 + 2^130의 차이를 인식하지 못하는 정도에 이르기 때문에 일반적인 방법으로는 정상적인 정수의 연산이 불가능하다. 따라서 매트랩에서 제공하는 자료형 int64를 이용하여 63bit 기반 ECC까지 구현 가능하였지만 연산 시간이 길어져서 실제 실험해보지는 못했다.

4-2. 진행과정에서의 문제점 및 해결방안

타원 곡선 위의 두 점 P, Q의 덧셈 연산은 P, Q를 잇는 직선과 타원 곡선의 P, Q가 아닌 교점 R의 x축 대칭점 S를 결과로 갖는다. 이 때 P, Q의 y좌표가 모두 0일 경우 P, Q가 아닌 점 R을 만들지 않거나, y좌표가 0인 R만을 만듦을 실험을 통해 알 수 있었다. 이러한 결과는 base point G의 선택과 공개키 Q의 연산에 영향을 미친다.

(1) Base point G의 경우, G의 y좌표가 0이면 위수 n이 매우 작아져 보안상 취약점을 갖게 된다. 따라서 base point G를 정할 때에는 G의 y좌표가 0이 되어서는 안된다.

(2) 공개키 Q의 연산에서는 Q의 y좌표가 0일 때, k*Q(=k*d*G)와 d*C1(=d*k*G)가 다른 값을 갖게 되어 정상적인 암호화/복호화가 되지 않는다. Q가 (i,0)일 때 k*Q와 d*C1이 다른 값을 가지게 되는 것은 y좌표가 0이면 타원 곡선의 성질을 생각할 때 점 Q에서의 접선이 x축에 수직이 되어 일반적인 상황에서의 (i*j)*P =i*(j*P) (i,j는 정수, P는 곡선위의 한 점)가 성립하지 않기 때문인 것으로 보인다.

(1)에서 나타난 문제의 경우, generate_G.m 함수를 사용해 base point G의 위수가 작아져 보안상 취약점을 갖지 않도록 하였다.

(2)에서 나타난 문제는 암호화 과정에서 y좌표가 0이면 어떤 식으로 그 값을 처리해야 할지에 대해 고민하였으나 아직 해결방안을 찾지 못하였다.

4-3. 프로그램 개선방안

(1) generate_g.m

Function Name	Calls	Total Time	Self Time*	Total Time Plot (dark band = self time)
public_parameter	1	5.457 s	0.004 s	
generate_G	1	5.417 s	0.005 s	
check_n	599	5.412 s	0.592 s	
ismember	79636	3.753 s	1.008 s	
ismember>ismemberR2012a	79636	1.934 s	1.932 s	
add_dot	79693	1.073 s	0.414 s	
partialMatchString	79635	0.811 s	0.696 s	
Inv	79618	0.659 s	0.659 s	

< 실행시간 측정 : generate_g.m 수정 전 >

Function Name	Calls	Total Time	Self Time*	Total Time Plot (dark band = self time)
public_parameter	1	2.845 s	0.004 s	
generate_G	1	2.807 s	0.004 s	
check_n	300	2.803 s	0.306 s	
ismember	39819	1.947 s	0.525 s	
ismember>ismemberR2012a	39819	1.000 s	0.998 s	
add_dot	39876	0.556 s	0.216 s	
partialMatchString	39818	0.422 s	0.362 s	
Inv	39838	0.340 s	0.340 s	

< 실행시간 측정 : generate_g.m 수정 후 >

generate_g.m을 수정하기 전의 public_parameter.m실행시간은 p=599일 때, 5.457s이다. 이는 p의 크기에 비해 매우 오랜 시간이 걸리는 것으로 프로그램의 효율성이 떨어진다고 볼 수 있다. 이 코드의 비효율성은 코드 내 generate_G 함수에서 많은 시간을 소모하기 때문인데, p에 따라 늘어나는 타원 곡선위의 모든 점에 대해 위수를 구하기 때문에 시간이 오래 걸릴 수 밖에 없다. 이 시간을 줄이기 위해 하나의 x에 대해 y1, y2가 존재하며, 이 둘의 위수가 같음을 이용하여 x좌표가 다른 점만을 검사하도록 하여 검사하는 점의 개수를 절반으로 줄

였다. 그 결과 generate_g.m의 실행시간이 절반가량 줄어들었고 public_parameter.m의 실행시간도 비슷하게 줄어들었다.

(2) multiple_dot.m

3-4-7의 multiple_dot.m의 코드를 보면 for문을 돌면서 단순히 점 P를 k번 더하고 있음을 확인할 수 있다. 이 때의 시간복잡도는 O(n)이다. 이를 개선할 수 있는 여러 방법들이 제시되어 있는데 대표적으로 double-and-add 라 불리는 방법이다. 예를 들어 27*P를 연산하고자 할 때 $200 = 2^7+2^6+2^3$ 임을 이용하여 2*2*2*(2*2*2*(2*P + P) + P)의 방식으로 연산량을 줄일 수 있다. 이 때의 시간복잡도는 O(log n)에 근접한다.

5. 역할 분담

팀원	역할	공동
강주연	예비보고서 - 과제 제목 및 선정이유 복호화 코드 구현 결과보고서 - 복호화 과정, 과제 결과 설명	자료조사 ECC 알고리즘 학습 상호 역할 피드백 예비/결과 보고서 수정
서지은	예비보고서(과제 내용) - 타원곡선암호(ECC) 소개 암호화 코드 구현1 결과보고서 - 암호화 과정, 과제 진행 과정의 문제점 및 해결 방법	
한시연	예비보고서(과제 내용) - 암/복호화 연구과정 예상 암호화 코드 구현2 결과보고서 - 암호화 과정, 결과물 개선 방안	
전호진 (팀장)	스케쥴 조정 및 역할 조정 예비/결과 보고서 초안 작성 PPT 디자인 발표자	

6. 참고 자료

1. Daniel R. L. Brown, *Recommended Elliptic Curve Domain Parameters,* Certicom Research, 2010 / https://goo.gl/BB4PPD

2. 심경아, *타원곡선 암호시스템 급부상,* 한국정보보호진흥원, 2001 / https://goo.gl/P5x6Sc

3. Hamish Silverwood, *A MATLAB Implementation of Elliptic Curve Cryptography*, Department of Mathematics and Statistics University of Canterbury, 2006-2007 / https://goo.gl/8ZnjNZ

4. Padma Bh1, D. Chandravathi, P.Prapoorna Roja, *Encoding And Decoding of a Message in the Implementation of Elliptic Curve Cryptography using Koblitz's Method,* International Journal on Computer Science and Engineering, 2010 / https://goo.gl/iAmFa7

5. Ravi Kishore Kodali & Prof. Narasimha Sarma, *Modified Koblitz Encoding Method for ECC*, National Institute of Technology, Warangal Department of E. and C. E., N.I.T., Warangal, India, 2013 / https://goo.gl/mt9Crc