# 2020년 알고리즘 기말 프로젝트 보고서

RSA 공개키 암호 해독

| 학 과 명 | 수학과 |
|---|---|
| 담당교수 | 김상일 교수님 |

| 조 명 : 스물넷이 막내인 조 (1조) | | | |
|---|---|---|---|
| 구 분 | 학 번 | 성 명 | 업무 분담 내용 |
| 조 장 | 201611108 | 김 동 현 | 연구내용 사전조사 및 문제 해결(30%), 보고서 작성(50%), PPT 제작(20%) |
| 팀 원 | 201511118 | 민 성 훈 | 연구내용 사전조사 및 문제 해결(20%), PPT 제작(30%), 발표(50%) |
| | 201511152 | 허 정 훈 | 연구내용 사전조사 및 문제 해결(20%), PPT 제작(30%), 발표(50%) |
| | 201711117 | 김 태 환 | 연구내용 사전조사 및 문제 해결(30%), 보고서 작성(50%), PPT 제작(20%) |

부산대학교
PUSAN NATIONAL UNIVERSITY

| | |
|---|---|
| **프로젝트 목적** | RSA 공개키 암호화 체계의 이해와 암호 해독의 프로그래밍 |
| **프로젝트 내용** | 암호 해독 코딩 |
| **코드 설명** | 구현한 함수들과 main함수에 대해 각 항목별로 주석을 달아 설명 |
| **실행 결과** | 후술 |
| **결과 분석** | 1. 구현 과정에서의 문제점과 해결방법<br>2. 구현 과정에서 배운 내용 설명<br>3. 함수에 대한 자세한 설명 |

# - 목 차 -

# Ⅰ. 암호(RSA)

암호란 타인에게 정보를 들키지 않고 보여주기 위한 장치이다. 물리적으로 정보가 넘어가지 않는 것도 중요하지만 정보가 넘어갔어도 그것이 무엇을 의미하는지 모르게만 할 수 있다면 충분하다. 따라서 암호는 그 정보를 감추는 기능을 한다. 암호의 종류는 무수히 많고 종류마다 성능도 다르지만 공통적인 특성이라 하면 평문을 암호화 하고 암호문을 다시 해독하는 복호화 과정을 갖는다. 즉, 암호는 평문을 암호문으로 바꾸는 '암호화 키'와 암호문을 평문으로 바꾸는 '복호화 키'를 갖는다. 암호를 분류하는 기준은 여러 가지가 있지만 가장 보편적인 방법으로 과거에 많이 쓰였던 '비밀키 암호 체계'와 현대에서 많이 쓰이는 '공개키 암호 체계'가 있다.

비밀키 암호 체계는 암호화 키와 복호화 키를 암호를 주고받는 사람만 알고 있는 체계이다.
이와 같은 전통적인 비밀키 암호 체계에서는 암호화 키와 복호화 키가 사실상 같아서 대칭키 암호 체계나 다름이 없다. 따라서 비밀키 암호화 체계에서는 암호화 키를 알아내면 복호화 키도 손쉽게 구할 수 있다. (쉽게 말해서, 암호화 키로부터 복호화 키를 구하는 간단한 함수가 존재한다고 할 수 있다.) 반면 공개키 암호화 체계는 암호화키와 복호화 키가 달라서 암호화 키를 구했다고 하더라도 복호화 키를 얻기 힘들다. 공개키 암호화 체계는 암호화 키와 복호화 키가 다르다는 점에서 비대칭키 암호 체계의 한 종류이다.

RSA는 대표적인 공개키 암호로서 Diffie와 Hellman의 공개키 암호 개념을 기반으로 MIT공대 연구팀 소속의 세 학자Rivest, Shamir, Adleman에 의해 1977년에 탄생되었고, RSA이름은 세 학자 이름의 머리글자를 따서 만든 명칭이다.
(사실 1975년에 영국의 수학자 Ellis, Cocks, Williamson이 RSA와 같은 암호 체계를 구상하였으나, 그들이 일하던 영국 정보기관인 정부통신본부(GCHQ)에서 기밀 정보로 분류하는 바람에 1997년까지 그들의 업적이 알려지지 않았다고 한다.)

RSA는 큰 수의 소인수분해가 매우 어렵다는 것을 기반으로 만들어 졌다. 즉 n=p*q일 때, p와 q로 n을 구하기는 쉬우나 n으로 p와 q를 찾기 힘들다는 소인수분해의 어려움을 이용하였다.
위의 성질을 통해 RSA는 n에 대한 잉여계에서 정의가 되는데 이 때, 공개키(e,N)와 비밀키(d,N)의 생성 방식은 큰 두 소수 p, q에 의존한다.

먼저 N=pq, L=$\phi(N)$이라고 하자. L의 기약잉여계에서 적당히 큰 수를 e로 설정한 후 d를 e의 L에 대한 역원으로 만든다. $d \quad e^{-1} (\mathrm{mod} \phi(N))$.

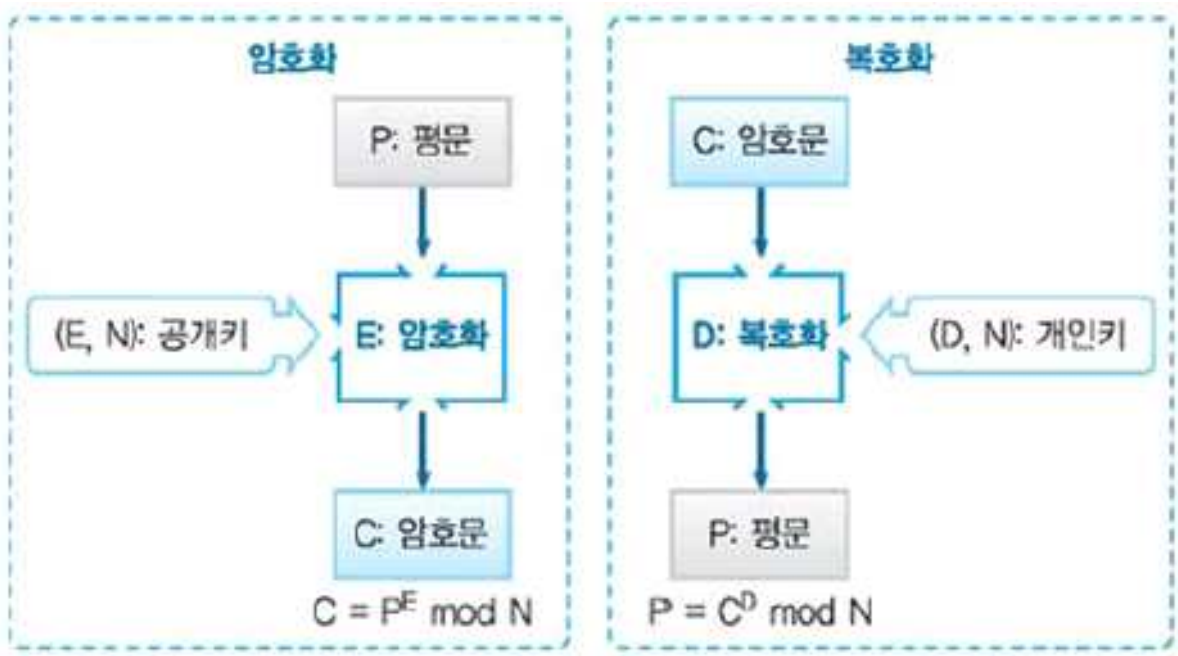
암호화: $P^e \equiv C (\mathrm{mod} N)$ where $0 \le C < N$

복호화: $C^d \quad P (\mathrm{mod} N)$ where $0 \le P < N$

위의 과정을 통해 암호화를 하거나 복호화를 진행한다. (C와 P는 각각 암호문과 평문이다.)

RSA 공개키 암호화 체계에서는 주어진 공개키를 통해 쉽게 비밀키를 알아낼 수 없다.
만약 우리가 암호를 해독하는 입장, 그리고 공개키(e,N)이 주어졌다고 하자. 분명히 d는 e의 $\phi(N)$에 대한 역원이므로 유일하게 존재한다. 하지만 $\phi(N)$을 구하기 위해선 N의 소인분해 형태인 p, q를 알아야한다. 충분히 큰 수의 소인수분해가 어렵기 때문에 이 과정은 불가능 할 것이다.

이처럼 공개키(e,N)이 주어졌다 하더라도 비밀키(d,N)은 두 소수 p와 q를 모르면 구할 수가 없다. 따라서 이 두 쌍의 키에서 공개키는 외부에 알려져도 상관이 없지만 비밀키가 유출이 되면 암호문을 해독할 수 있기 때문에 절대로 알려지면 안 된다.




우리의 과제는 비밀키가 주어진 상태에서 암호문을 법 N에 대한 연산을 통해 평문으로 바꾸는 해독 과정이다. 우리는 주어진 N(=817)에 대한 두 소인수(p=19, q=43)를 알고 있다. 따라서 후술하는 정수론 개념을 통해 코딩을 진행했고, 소인수가 주어졌을 때와 그렇지 않을 때의 경우의 연산 횟수를 비교하는 과정을 진행해보았다.

# Ⅱ. 정수론

다음은 우리의 해독 코딩을 매트랩을 통해 가능하게 해줄 정리들이다.

1. 오일러 정리/페르마 소정리

## · 오일러 정리

$gcd(a,n)=1$ 이면, $a^{\phi(n)} \equiv 1 \pmod n$ 이다.

여기서 $\phi(n)$은 오일러의 $\phi$ 함수이다.

「Recall 기약잉여계 (a reduced residue system modulo n)

정수 $a_1, a_2, \cdots, a_t$가 n과 서로소이고, 임의의 정수 x가 n과 서로소이면,

이 정수 x는 $a_1, a_2, \cdots, a_t$ 중의 오직 하나의 정수와 합동이 될 때,

$\{a_1, a_2, \cdots, a_t\}$를 법 n의 기약잉여계라 한다.」

**Pf)** Suppose that $\{r_1, r_2, \cdots, r_{\phi(n)}\}$ is a reduced residue system modulo n.

Then $gcd(a,n)=1$ and $gcd(r_i, n)=1$.

So, $\forall i = 1, 2, \cdots, \phi(n)$, $gcd(ar_i, n)=1$.

By definition of a reduced residue system modulo n,

$\forall i$, $\exists k \in \{1, 2, \cdots, \phi(n)\}$ s.t. $ar_i \equiv r_k \pmod n$

Claim: $\nexists k \in \{1, 2, \cdots, \phi(n)\}$ s.t. $r_k \equiv ar_i \pmod n$ and $r_k \equiv ar_{i'} \pmod n$

(i.e., $\forall k \in \{1, 2, \cdots, \phi(n)\}$, $\exists! i \in \{1, 2, \cdots, \phi(n)\}$ s.t. $r_k \equiv ar_i \pmod n$)

Pf) Suppose that $ar_i \equiv r_j \pmod n$ and $ar_{i'} \equiv r_j \pmod n$.

Then $ar_i \equiv ar_{i'} \pmod n$ i.e., $r_i \equiv r_{i'} \pmod n$

Since $\forall p, q \in \{1, 2, \cdots, \phi(n)\}$ with $p \neq q$, $r_p \not\equiv r_q \pmod n$, $r_i = r_{i'}$.

Thus $\{ar_1, ar_2, \cdots, ar_{\phi(n)}\}$ is also a reduced residue system modulo n.

Since $(ar_1)(ar_2)\cdots(ar_{\phi(n)}) \equiv r_1 r_2 \cdots r_{\phi(n)} \pmod n$, $a^{\phi(n)} r_1 r_2 \cdots r_{\phi(n)} \equiv r_1 r_2 \cdots r_{\phi(n)} \pmod n$

Since $gcd(r_1 r_2 \cdots r_{\phi(n)}, n) = 1$, $a^{\phi(n)} \equiv 1 \pmod n$ ∎

## · 페르마 소정리

P가 소수이고, $p \nmid a$ 이면 $a^{p-1} \equiv 1 \pmod p$ 이다.

이 정리는 오일러 정리의 특수한 경우이다.

2. 나눗셈 정리

## • 나눗셈 정리 (Division Algorithm)

$a$가 양의 정수이고, $b$가 임의의 정수이며, 다음 식을 만족하는 정수 $q$와 $r$은 유일하게 존재한다.

$$b = qa + r, \quad 0 \le r < a$$

**Pf)** i) By Archimedian Principle, $b \ge 0 \Rightarrow \forall a \in \mathbb{Z}^+, \exists n \in \mathbb{Z}^+$ s.t. $na > b$

Suppose that $n = q+1$ satisfying $na > b$. Then $(q+1)a > b \ge qa$.

Since $b \ge qa$ and $(q+1)a = qa + a > b$,

$r := b - qa \Rightarrow r = b - qa \ge 0$ and $r = b - qa < a$

Similarly, $b < 0 \Rightarrow$ it also holds.

Thus, $b = qa + r$, $0 \le r < a$.

ii) Assume that $b = qa + r = q_1 a + r_1$, $0 \le r, r_1 < a$. (W.L.O.G. $r_1 \le r$)

Then $0 \le r - r_1 < a$. $\cdots$ ⊛

Since $(q_1 - q)a = r - r_1$, $a \mid r - r_1$ and $r - r_1 > 0 \Rightarrow a \le r - r_1$.

It contradicts ⊛

Thus $r - r_1 = 0$ i.e., $r = r_1$ and $q = q_1$

Hence $\forall a \in \mathbb{Z}^+, b \in \mathbb{Z}, \exists! q, r \in \mathbb{Z}$ s.t. $b = qa + r$, $0 \le r < a$. ∎

3. 확장된 유클리드 정리

## • 확장된 유클리드 호제법

$\forall a, b \in \mathbb{Z}, \exists x, y \in \mathbb{Z}$ s.t. $ax + by = gcd(a, b)$.

「Recall 유클리드 호제법

$\forall a, b, r, q \in \mathbb{Z}$ with $a \ge b$, $b \ge r \ge 0$, $a = bq + r \Rightarrow gcd(a, b) = gcd(b, r)$」

**Pf)** By 유클리드 호제법, $r_1 = a - q_1 b$ $\cdots$ ⓐ (a와 b의 선형결합)

Similarly, $r_2 = b - q_2 r_1$ $\cdots$ ⓑ

By ⓐ, ⓑ, $r_2 = b - q_2(a - q_1 b) = -q_2 a + (1 + q_1 q_2) b$ (a와 b의 선형결합)

Continuing this process, $\exists x, y \in \mathbb{Z}$ s.t. $r_n = ax + by$

By 유클리드 호제법, $r_n = gcd(a, b)$

Hence, proof is done! ∎

4. 중국인의 나머지 정리

## • 중국인의 나머지 정리

양의 정수 $m_1, m_2, \dots, m_t$ 가 서로 소라 하자. 즉, $\gcd(m_i, m_j) = 1,\ i \neq j,\ 1 \le i, j \le t$.

또한, $b_1, b_2, \dots, b_t$ 는 임의의 정수라면 다음 연립합동식은 하나의 연립해를 가진다.

$$\begin{cases} x \equiv b_1 \pmod{m_1} \\ x \equiv b_2 \pmod{m_2} \\ \vdots \\ x \equiv b_t \pmod{m_t} \end{cases}$$

그리고 그 연립해는 $m_1, m_2, \dots, m_t$ 에 대하여 유일한 해가 된다.

(즉, $y$가 또 다른 하나의 해라면 $x \equiv y \pmod{m_1 m_2 \cdots m_t}$이다.)

**Pf)** i) Let $y_i \equiv 1 \pmod{m_i}$ and $y_i \equiv 0 \pmod{m_j}$ for $i, j = 1, 2, \dots, t$ with $i \neq j$.

Then $x = y_1 b_1 + y_2 b_2 + \cdots + y_t b_t$.

Since $y_1 \equiv 0 \pmod{m_i}$ and $m_i$ are relative prime, $2 \le i \le t$, $m_2 m_3 \cdots m_t \mid y_1$.

Let $m_i' = \frac{m_1 m_2 \cdots m_t}{m_i}$. Since $\gcd(m_i, m_i') = 1$, $m_i'^{*} m_i' \equiv 1 \pmod{m_i}$

Let $y_i = m_i'^{*} \cdot m_i'$. Then $x = m_1'^{*} m_1' b_1 + m_2'^{*} \cdot m_2' \cdot b_2 + \cdots + m_t'^{*} \cdot m_t' \cdot b_t$.

Since $m_1 \mid m_i'$, $2 \le i \le t$, $m_i'^{*} m_i' b_i \equiv 0 \pmod{m_1}$

Since $m_1'^{*} m_1' \equiv 1 \pmod{m_1}$ and $m_1'^{*} m_1' b_1 \equiv b_1 \pmod{m_1}$,

$x \equiv b_1 + 0 + \cdots + 0 \equiv b_1 \pmod{m_1}$

Similar argument holds $x \equiv b_i \pmod{m_i}$ for $1 \le i \le t$.

$\therefore x$는 위의 연립합동식의 해가 된다.

ii) Let $z \equiv b_i \pmod{m_i}$ for $1 \le i \le t$. Then $x \equiv z \pmod{m_i}$.

Since $m_i$ are relative prime, $\forall i$, $m_i \mid x - z$ i.e., $m_1 m_2 \cdots m_t \mid x - z$.

Thus $x \equiv z \pmod{m_1 m_2 \cdots m_t}$

$\therefore x$는 유일한 해가 된다. ∎

# Ⅲ. 코딩

## 1. Decoding.m

```
clear; clc;                                          %기존 메모리 삭제
private_key=[605 817];                               %개인키

%수학 이론 미적용(Solve1.m 이용)
fid = fopen("rsa.txt", 'r');                         %암호문 읽기
tic;
while ~feof(fid)
      pw=str2num(fgetl(fid));                        %암호문을 숫자 벡터로 전환
      pw=Solve1(pw, private_key);                    %각 암호에 대한 해독
      fprintf('%s₩n',char(pw));                      %해독문 출력
end
fprintf('%s', "수학 이론 미적용 시 ");
toc;                                                 %수학 이론 미적용 시 해독시간 측정
fclose(fid);

fprintf('₩n');

%수학 이론 적용(Solve2.m 이용)
fid = fopen("rsa.txt", 'r');                         %암호문 읽기
Factors=factor(private_key(2));
%N을 두 개의 소수 p, q로 나누어 이를 벡터로 저장
%일반적인 rsa의 경우 N이 크기 때문에 연산시간 증가 등의 문제가 발생

tic;
inver=Inverse(Factors);              %Z_p에서 q의 역원과 Z_q에서 p의 역원을 각각 구함
while ~feof(fid)
      pw=str2num(fgetl(fid));                        %암호문을 숫자 벡터로 전환
      pw=Solve2(pw, private_key, Factors, inver);    %각 암호에 대한 해독
      fprintf('%s₩n',char(pw));                      %해독문 출력
end
fprintf('%s', "수학 이론 적용 시 ");
toc;                                                 %수학 이론 적용 시 해독시간 측정
fclose(fid);
```

2.

(1) Solve1.m

```
%암호 행렬과 개인키를 받아 해독문을 반환하는 함수
function c=Solve1(A, private_key)
    c=A;
    for n=1:length(A)
        for i=1:private_key(1)-1
            c(n)=mod(c(n)*A(n),private_key(2));
        end
    end
end
```

- $a^n \equiv \square \pmod{N}$이라 할 때,
  => $a \times a \equiv x_1 \pmod{N}$
  => $x_1 \times a \equiv x_2 \pmod{N}$
  => $x_2 \times a \equiv x_3 \pmod{N}$
  => …… (이를 총 n-1회 반복)
  => $\therefore\ a^n \equiv x_{n-1} \pmod{N}$

- 위의 식에서 □를 구하기 위의 연산이 약 600번 반복된다.

- 단, 매트랩(MATLAB) 프로그램을 돌리기 위해서는 N(=private_key(2))의 값이 $10^{14}$ 자리수 이내의 수를 입력해야 한다.

(2) Solve2.m

```
1  %암호 행렬과 개인키를 받아 해독문을 반환하는 함수
2  function c=Solve(A, private_key, Factors, inver)
3      for n=1:length(A)
4          %중국인의 나머지 정리를 사용하기 위해 p,q에 대한 연립합동식을 계산
5          m=[mod(A(n),Factors(1)) mod(A(n),Factors(2))];
6          x=m;
7          %큰 수의 경우 근삿값으로 계산하여 오차가 발생하는 오류를 잡기 위해
8           반복 작업의 결과를 저장할 변수 생성
9          %(ex) 183^605 (mod 19)의 계산을 12^11 (mod 19)로 수의 크기를 줄여도
10          여전히 값이 크다.
11          %(cf) 위의 과정은 페르마의 소정리를 이용하여 수를 줄인 경우이다.
12          %하지만 이 과정을 12 (mod 19)를 구하여 이를 11번 반복하는 형식으로
13           값의 크기를 줄여 근삿값에 의해
14          %오류가 발생하는 것을 방지할 수 있다.
15          for i=1:2                    %위의 설명과 같이 반복 작업 실행
16              for j=1:(mod(private_key(1), Factors(i)-1)-1)
17                  x(i)=mod(x(i)*m(i), Factors(i));
18              end
19          end
20          %구한 역원을 통해 유일해를 유추
21          c(n)=mod(x(1)*Factors(2)*inver(2)+x(2)*Factors(1)*inver(1), private_key(2));
22      end
23  end
```

- 5 : $a^n \pmod{x}$의 식에서 $x$의 값을 구하는 과정에서 a의 값이 상당히 크다면 비교적 연산이 느려질 수밖에 없다. 이때 $N=p\times q$라고 하면 **나눗셈 정리**를 통하여 $a^n \equiv b^n \pmod{p}\,(a>b)$, $a^n \equiv c^n \pmod{q}\,(a>c)$를 만족하는 b와 c값을 구함으로써 연산속도를 향상시킬 수 있다.
(연산의 크기를 줄임으로써 연산속도를 향상시킴)

- 15: $a^n \equiv b^n \pmod{p}\,(a>b)$의 식에서 n의 값이 상당히 크다면 비교적 연산이 느려질 수밖에 없다. 이때 **페르마 소정리**(**오일러의 정리**의 특수한 경우)를 통하여 $n \equiv k \pmod{\phi(p)}$를 만족하는 적당히 작은 k를 구함으로써 연산속도를 향상시킬 수 있다.
(연산의 횟수를 줄임으로써 연산속도를 향상시킴)

- 21: $a^n \equiv b^n \pmod{p}\,(a>b)$와 $a^n \equiv c^n \pmod{q}\,(a>c)$에 대하여 **중국인의 나머지 정리**를 이용해서 $a^n \equiv x \pmod{N}$ $(N=p\times q)$를 만족하는 $x$값을 구한다. (이때, a=암호문, $x$=평문)

## 3. Inverse.m

초기 조건을 대입하여 확인할 수 있으며, $i > 1$일 때 참이라 가정하고 수학적 귀납법을 사용하면 다음과 같다.

$$r_{i+1} = r_{i-1} - r_i q_i$$

$$= (as_{i-1} + bt_{i-1}) - (as_i + bt_i)q_i$$

$$= a(s_{i-1} - s_i q_i) + b(t_{i-1} - t_i q_i)$$

$$= as_{i+1} + bt_{i+1}$$

따라서 $r_{i+1} = 0$일 때, $as_i + bt_i = r_i$는 $sa + tb = gcd(a, b)$가 된다.

위의 사실을 바탕으로 아래와 같이 코딩해보았다.

```
1  function inver=Inverse(Factors)
2      s(1) = 1; s(2) = 0;
3      t(1) = 0; t(2) = 1;
4      r(1) = Factors(1); r(2) = Factors(2);
5      q(1) = 0; q(2) = fix(r(1)/r(2));
6      ind = 3;
7      while true
8          r(ind) = mod(r(ind-2), r(ind-1));
9          if r(ind) == 0      %r(ind-1)이 gcd(a,b)를 만족하는 경우 반복 작업을 중지한다.
10             break;
11         end
12         q(ind) = fix(r(ind-1)/r(ind));
13       %확장된 유클리드 호제법을 반복 적용하여 s와 t를 구한다
14         s(ind) = s(ind-2) - s(ind-1)*q(ind-1);
15         t(ind) = t(ind-2) - t(ind-1)*q(ind-1);
16         ind = ind + 1;
17     end
18     %s는 p에 대한 Z_q에서의 역원, t는 q에 대한 Z_p에서의 역원이 된다.
19     inver = [s(ind-1) t(ind-1)];
20 end
```

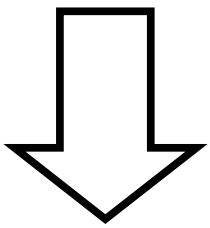
- 중국인의 나머지 정리를 이용하여 합동방정식의 해를 구하는 과정에서 필요한 mod p에 대한 q의 역원과 mod q에 대한 p의 역원을 확장된 유클리드 알고리즘을 통해 구한다.
  ($N = 817 = 19 \times 43$  By 확장된 유클리드, $\gcd(19, 43) = (-9) \times 19 + 4 \times 43 \Rightarrow s = -9, t = 4$)

4. Result

## Q. 다음 암호를 해석하세요.

The public key is <e, N> = <5, 817>
The private key is <d, N> = <605, 817>

Ciper Text is
183 320 669 447 317 20 242 46 66 288 20 669 572 242 242
737 224 779 242 126 779 81 317 184 81 20 184 242 104 81 317 692 572 81 20 20 242 641 288 81 20 242 288 572 242 126 288 758 288 572 126 242 224 519 544
183 320 81 242 447 669 20 184 242 74 81 779 184 317 288 572 242 104 317 600 242 184 669 242 20 224 74 74 81 81 66 242 288 20 242
317 641 104 317 600 20 242 184 669 242 184 779 600 242 26 224 20 184 242 669 572 81 242 447 669 779 81 242 184 288 447 81 544



명령 창

```
Thomas Edison
Our greatest weakness lies in giving up.
The most certain way to succeed is
always to try just one more time.
수학 이론 미적용 시 경과 시간은 0.022041초입니다.

Thomas Edison
Our greatest weakness lies in giving up.
The most certain way to succeed is
always to try just one more time.
수학 이론 적용 시 경과 시간은 0.001498초입니다.
```

- Solve1(N에 대한 소인수를 모를 때) 함수를 사용했을 때보다 정수론 개념을 활용하여 코딩한 Solve2(N에 대한 소인수를 알 때) 함수를 사용했을 때 처리속도가 더 빠르다는 것을 알 수 있다.