

컴퓨터보안 보고서

실습과제 #7

강좌 명: 컴퓨터보안

교수님: 정준호 교수님

학과: 컴퓨터공학과

학년: 3학년

학번: 2017112138

이름: 정여준

문제

이산대수가 어떻게 암호화 알고리즘에 적용되는지 조사 및 분석한다.

이산대수문제는 보통 $a^x = b$ 를 만족하는 x 를 찾는 것이다. 이산대수문제를 이해하기 위해서 먼저 이해해야 하는 수학적 개념이 있다. 일단 집합과 그 집합의 원소들이 취하는 연산이 함께 있는 개념에 대해서 알아야 한다. 예를 들어 $Z_n = \{0, 1, 2, \dots, n-1\}$ 이라는 집합이 있다고 하면 집합 Z_n 과 덧셈 연산을 $(Z_n, +)$ 이렇게 함께 표현한다. 여기서 Z_n 이라는 집합은 좀 복잡한 집합인데 구체적인 예를 제시하면 $n = 6$ 이라고 가정해보자. $Z_6 = \{0, 1, 2, 3, 4, 5\}$ 이 되는데 여기서 $0, 1, 2, 3, 4, 5$ 의 각 원소들은 말 그대로 숫자를 의미하는 것이 아니다. 사실 Z_6 은 $\{[0]_6, [1]_6, [2]_6, [3]_6, [4]_6, [5]_6\}$ 인 집합이다. 여기서 $[0]_6, [1]_6$ 과 같은 것은 $\text{mod } n$ 에 대한 동치류로 $[a]_n = \{a + kn \mid k \in \mathbb{Z}\}$ 즉, n 으로 나눈 나머지가 a 인 정수들의 집합인 것이다. 그렇기 때문에 $[0]_6$ 은 6으로 나눈 나머지가 0인 정수들을 의미한다. 따라서 여기에는 $-12, -6, 0, 6, 12$ 등이 있다. 이런 식으로 $[5]_6$ 까지 표현하게 되면 사실 상 모든 정수를 $\text{mod } 6$ 이라는 개념으로 그룹을 지어 표현한 것이 된다. 따라서 앞으로 Z_n 이라고 하면 $\text{mod } n$ 에 대한 동치류들의 집합이라고 이해하면 된다. 그룹 중에 $(Z_n, +)$ 도 있지만 $(Z_n^*, *)$ 이라는 곱셈 그룹도 있다. $Z_n^* = \{[a]_n \in Z_n \mid \gcd(a, n) = 1\}$ 즉, Z_n^* 은 Z_n 의 원소들 중에서 n 과 서로소인 a 들만 모은 것이다. Z_n^* 은 $\text{mod } n$ 에 대한 동치류들의 집합인 것 까지는 Z_n 과 똑같은데, 이 집합의 연산은 곱셈이라는 것, 이 동치류들이 $\text{mod } n$ 에 대해서 n 과 서로소라는 것을 알고 있으면 된다. 이 것을 활용해서 암호화 알고리즘에 적용하는 것이다.

이산대수가 암호화 알고리즘에 적용되는 방법은 다음과 같다. 일단 소수인 p 와 q 를 선택한다. 여기서 이 둘을 곱한 값 즉, $p \times q$ 가 N 의 값이 된다. 이후 $p-1$ 과 $q-1$ 의 최소공배수가 L 이 된다. 즉, $L = \text{lcm}(p-1, q-1)$ 이다. 이후에 E 는 L 과 최대공약수가 1이 되는 임의의 수이다. 즉 $\gcd(E, L) = 1$ 이 되는 수 중 하나를 선택하면 된다. 이후에 D 는 $E \times D \text{ mod } L = 1$ 이 되는 D 를 찾으면 된다. 여기서 공개 키는 (E, N) 이 되고 개인 키는 (D, N) 이 된다.

즉, 암호화할 때는 $\text{평문}^E \text{ mod } N$ 한 것이 암호문이 되고 복호화 할 때는 $\text{암호문}^D \text{ mod } N$ 을 한 것이 평문이 된다. 암호해독자가 알고 있는 것은 암호문과 공개키 E 와 N 이다. 하지만 암호해독자는 개인 키 D 와 키 쌍을 만든 p, q, L 을 알 수 없다. 그렇게 때문에 암호문으로부터 평문을 구하기 위해서 이산대수 문제를 풀어야 한다. 하지만 이산대수 문제는 아직까지도 빠르게 구하는 방법을 찾지 못했다. 그렇다고 전사공격을 하면 D 의 후보 수를 이용해서 모두 복호화를 시도해보겠지만 D 의 비트 수가 크면 클수록 어려움이 생긴다. 즉, 비트 수가 어느정도 큰 D 에 대해서 복호화를 전사공격으로 진행하는 것은 현실 적으로 불가능하다. 그리고 해독자가 E 와 N 을 안다고 해서 그 수들을 이용해서 D 를 구하는 것도 p 와 q 와 L 을 모르기 때문에 해독할 수 없다. $N = p \times q$ 라는 것을 이용해서 p 와 q 를 구하는 것도 현실적으로 힘든데 이는 아직까지도 큰 수를 빠르게 소인수분

해를 할 수 있는 방법이 없기 때문이다. 그렇다고 해서 이런 이산대수를 사용한 공개키 암호가 단점이 없는 것은 아니다. 만약 의사 난수 생성기의 품질이 안 좋다면 암호해독자가 p 와 q 를 추측할 수 있다. 그렇기 때문에 성능이 좋은 의사 난수 생성기를 사용해야한다. 또한 중간자 공격에 취약한데 중간자 공격이란 공격자가 수신자의 공개키를 저장한 후에 수신자의 공개키 대신 자신의 공개키로 송신자에게 전달한다. 그러면 송신자는 공격자의 공개키를 이용해서 암호화할 것이다. 이것을 다시 공격자가 받아 자신의 키로 복호화하고 위조문을 만들어서 수신자의 공개키로 다시 암호화한다. 그리고 위조된 암호문을 수신자에게 전달한다. 따라서, 공개키 암호 알고리즘은 이런 방식의 공격에 취약하다. 또한 공개키 암호는 대칭 암호에 비해서 몇 백배나 느리다.

$k^x \bmod N = 1$ 에서 N 을 입력 받아 성립하는 x 를 찾는 프로그램을 작성한다.

- 단, K 는 N 과 서로소가 되는 값 후보군(1은 제외) 중 랜덤하게 선택하도록 한다.
- 단, N 은 소수1 * 소수2로 만들어지는 값이며, 소수1과 소수2는 다른 값이다.
- 즉, $2*3$, $3*11$ 등이 될 수 있다.

<소스코드>

```
#include<iostream>
#include<cstdlib>
#include<ctime>
using namespace std;

int gcd(int a, int b) //최대 공약수 구하는 함수
{
    int temp, x = a, y = b;
    while (y != 0)
    {
        temp = y;
        y = x % y;
        x = temp;
    }
    return x;
}

int randPrime(int n) // 랜덤한 서로소 생성
{
    int a = rand();
    while (gcd(a, n) != 1)
    {
        a = rand();
    }
    return a;
}

int main()
{
    cin.tie(0);
    clock_t start, end;
    int N = 0;
    cout << "정수 N 입력: ";
    cin >> N;
    start = clock();
    for (int i = 0; i < 100; i++) { //100회 반복시행
        int k = randPrime(N);
        int x = 1;
        for (int result = k % N; result != 1; ++x) //x를 찾기 위한 반복문
        {
            result = (result + k) % N;
        }
    }
    end = clock();
    cout << "실행 시간: " << end - start << endl;
}
```

```
        cout << k << "^" << x << " mod " << N << "= 1" << endl;
    }
    end = clock();
    cout << "100회 평균 시간: "<<(float)((end - start) / 100) / CLOCKS_PER_SEC; //평균
시간 구하기 위해
    return 0;
}
```

N을 10진수를 기준으로 두 자리, 세 자리, 네 자리, ... ,최대 10자리까지 늘려가며 각 케이스 별로 100회 테스트하여 X를 찾는 데까지 소요된 시간을 측정 및 분석한다.

두 자리: $3*7 = 21$

세 자리: $23*29 = 667$

네 자리: $31*37 = 1147$

다섯 자리: $101*103 = 10403$

여섯 자리: $331*337 = 111547$

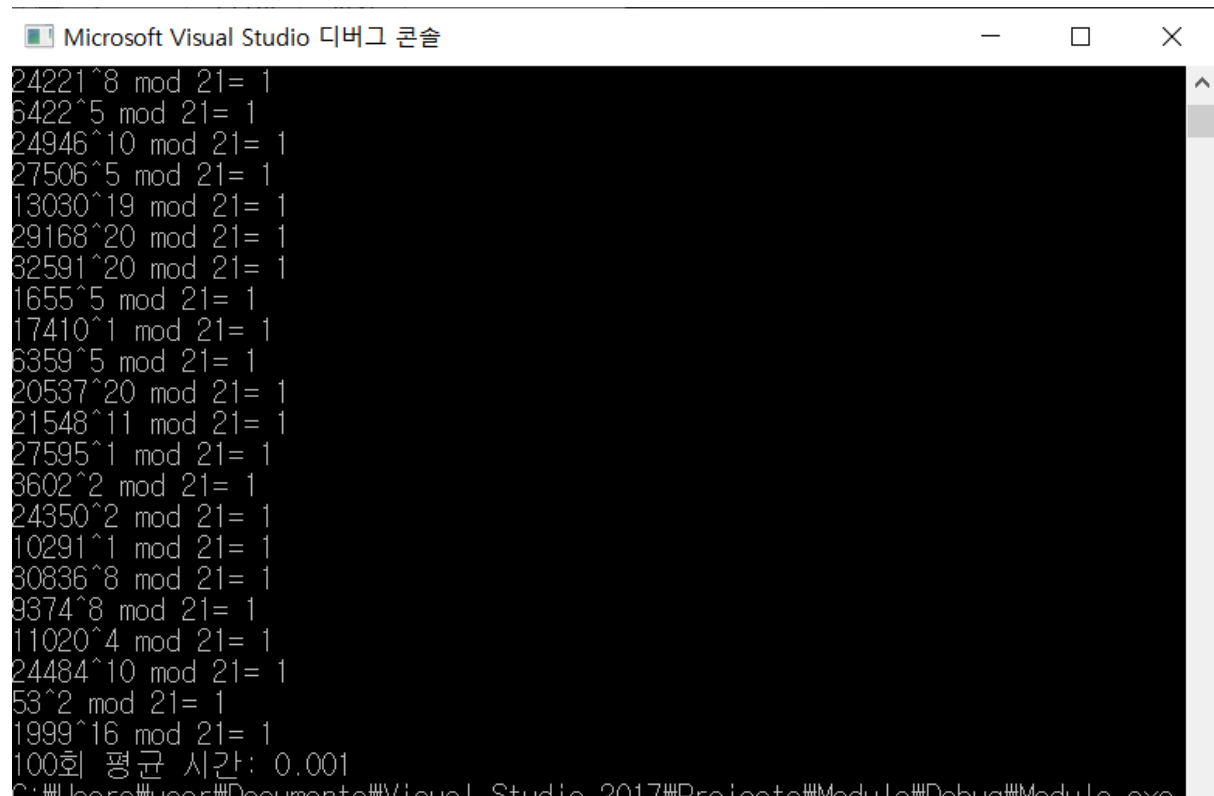
일곱 자리: $1009*1013 = 1022117$

여덟 자리: $3331*3343 = 11135533$

아홉 자리: $10007*10009 = 100160063$

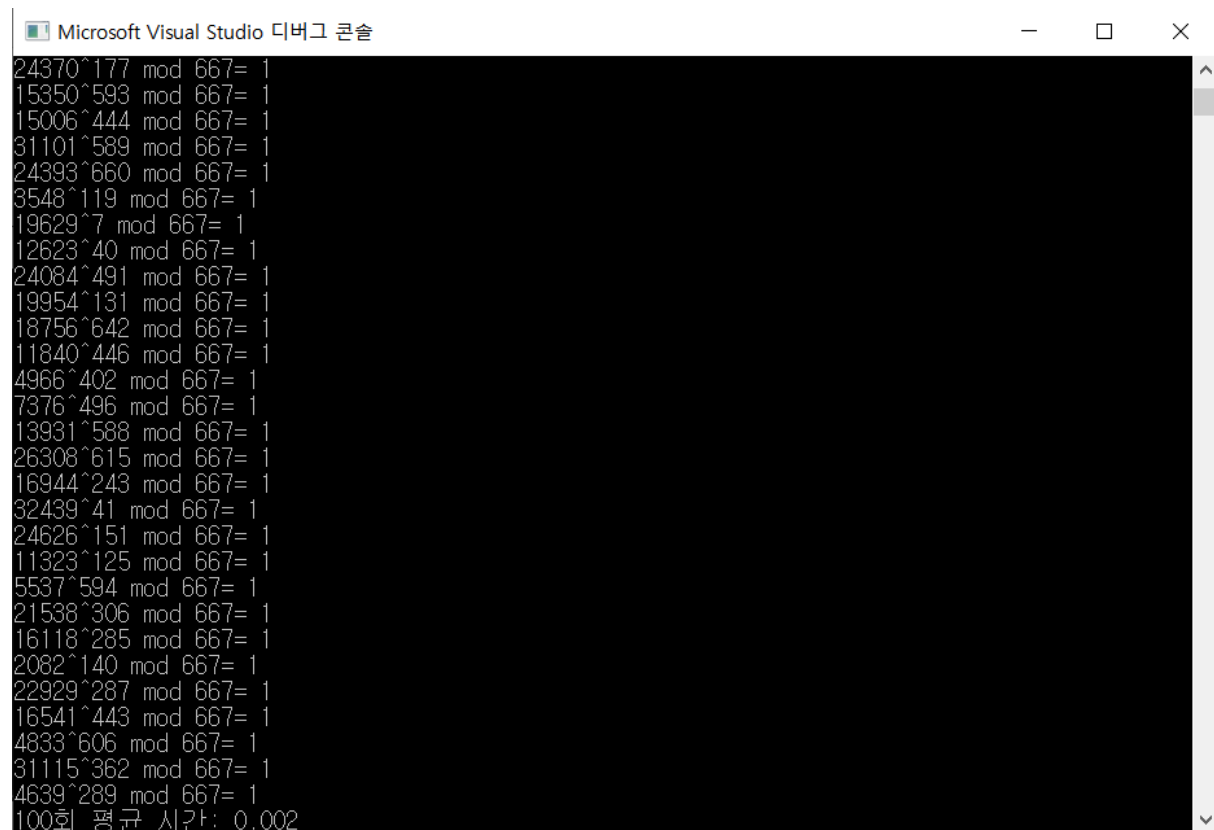
열 자리: $33331 * 33343 = 1111355533$

N = 21



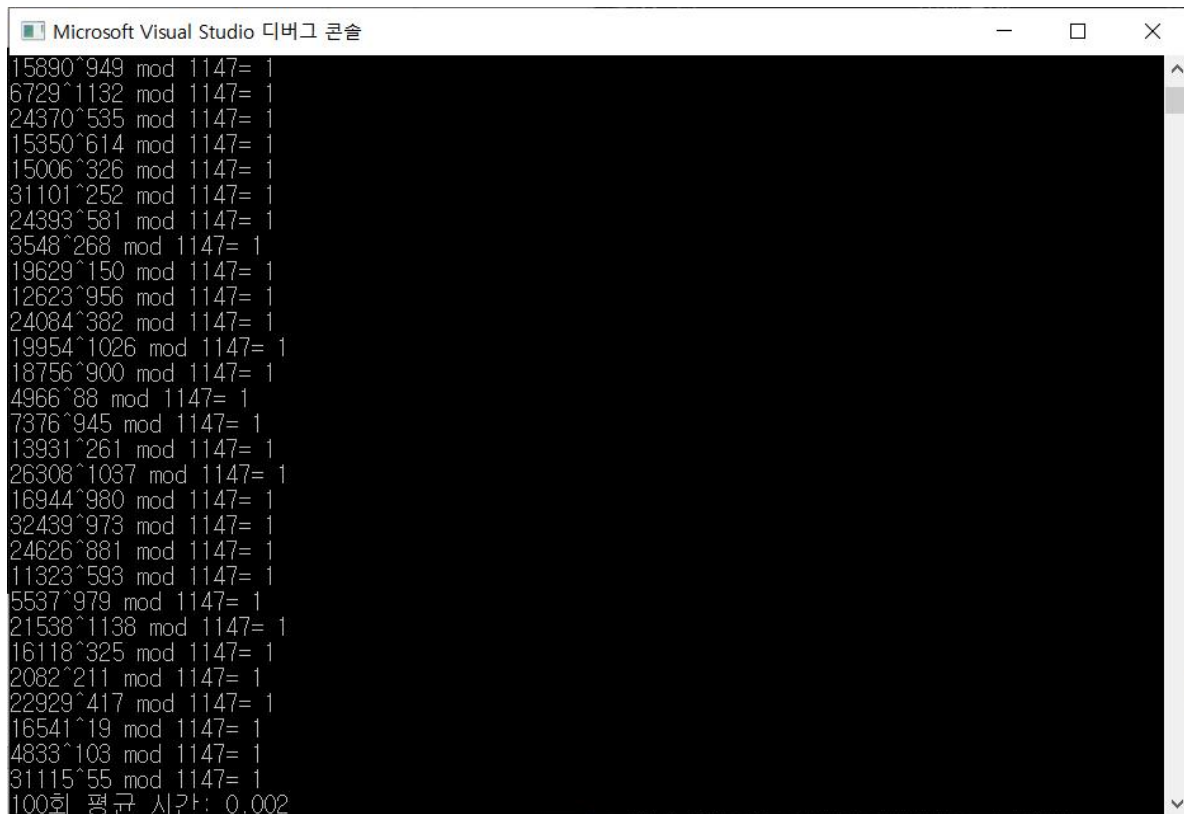
```
Microsoft Visual Studio 디버그 콘솔
24221^8 mod 21= 1
6422^5 mod 21= 1
24946^10 mod 21= 1
27506^5 mod 21= 1
13030^19 mod 21= 1
29168^20 mod 21= 1
32591^20 mod 21= 1
1655^5 mod 21= 1
17410^1 mod 21= 1
6359^5 mod 21= 1
20537^20 mod 21= 1
21548^11 mod 21= 1
27595^1 mod 21= 1
3602^2 mod 21= 1
24350^2 mod 21= 1
10291^1 mod 21= 1
30836^8 mod 21= 1
9374^8 mod 21= 1
11020^4 mod 21= 1
24484^10 mod 21= 1
53^2 mod 21= 1
1999^16 mod 21= 1
100회 평균 시간: 0.001
C:\Users\user\Documents\Visual Studio 2017\Projects\Module\Debug\Module.exe
```

N = 667



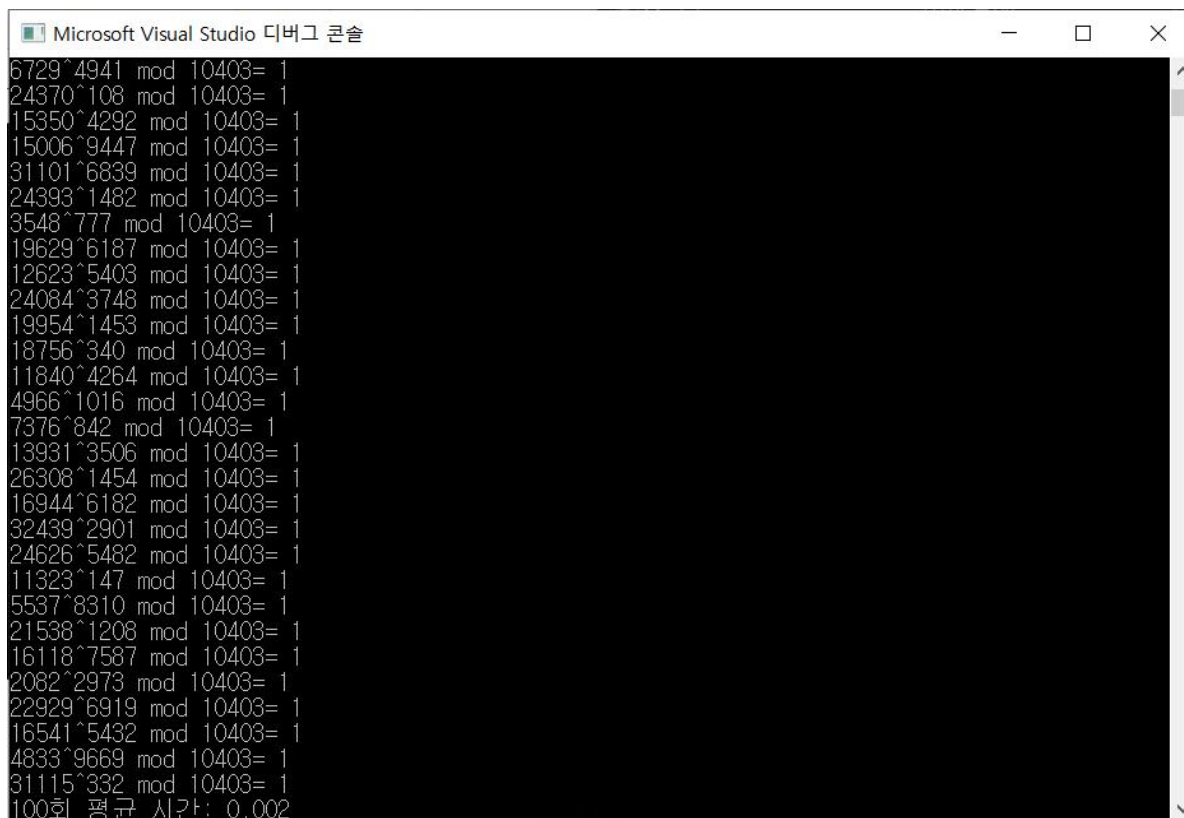
```
Microsoft Visual Studio 디버그 콘솔
24370^177 mod 667= 1
15350^593 mod 667= 1
15006^444 mod 667= 1
31101^589 mod 667= 1
24393^660 mod 667= 1
3548^119 mod 667= 1
19629^7 mod 667= 1
12623^40 mod 667= 1
24084^491 mod 667= 1
19954^131 mod 667= 1
18756^642 mod 667= 1
11840^446 mod 667= 1
4966^402 mod 667= 1
7376^496 mod 667= 1
13931^588 mod 667= 1
26308^615 mod 667= 1
16944^243 mod 667= 1
32439^41 mod 667= 1
24626^151 mod 667= 1
11323^125 mod 667= 1
5537^594 mod 667= 1
21538^306 mod 667= 1
16118^285 mod 667= 1
2082^140 mod 667= 1
22929^287 mod 667= 1
16541^443 mod 667= 1
4833^606 mod 667= 1
31115^362 mod 667= 1
4639^289 mod 667= 1
100회 평균 시간: 0.002
```

N = 1147



```
Microsoft Visual Studio 디버그 콘솔
15890^949 mod 1147= 1
6729^1132 mod 1147= 1
24370^535 mod 1147= 1
15350^614 mod 1147= 1
15006^326 mod 1147= 1
31101^252 mod 1147= 1
24393^581 mod 1147= 1
3548^268 mod 1147= 1
19629^150 mod 1147= 1
12623^956 mod 1147= 1
24084^382 mod 1147= 1
19954^1026 mod 1147= 1
18756^900 mod 1147= 1
4966^88 mod 1147= 1
7376^945 mod 1147= 1
13931^261 mod 1147= 1
26308^1037 mod 1147= 1
16944^980 mod 1147= 1
32439^973 mod 1147= 1
24626^881 mod 1147= 1
11323^593 mod 1147= 1
5537^979 mod 1147= 1
21538^1138 mod 1147= 1
16118^325 mod 1147= 1
2082^211 mod 1147= 1
22929^417 mod 1147= 1
16541^19 mod 1147= 1
4833^103 mod 1147= 1
31115^55 mod 1147= 1
100회 평균 시간: 0.002
```

N = 10403



```
Microsoft Visual Studio 디버그 콘솔
6729^4941 mod 10403= 1
24370^108 mod 10403= 1
15350^4292 mod 10403= 1
15006^9447 mod 10403= 1
31101^6839 mod 10403= 1
24393^1482 mod 10403= 1
3548^777 mod 10403= 1
19629^6187 mod 10403= 1
12623^5403 mod 10403= 1
24084^3748 mod 10403= 1
19954^1453 mod 10403= 1
18756^340 mod 10403= 1
11840^4264 mod 10403= 1
4966^1016 mod 10403= 1
7376^842 mod 10403= 1
13931^3506 mod 10403= 1
26308^1454 mod 10403= 1
16944^6182 mod 10403= 1
32439^2901 mod 10403= 1
24626^5482 mod 10403= 1
11323^147 mod 10403= 1
5537^8310 mod 10403= 1
21538^1208 mod 10403= 1
16118^7587 mod 10403= 1
2082^2973 mod 10403= 1
22929^6919 mod 10403= 1
16541^5432 mod 10403= 1
4833^9669 mod 10403= 1
31115^332 mod 10403= 1
100회 평균 시간: 0.002
```

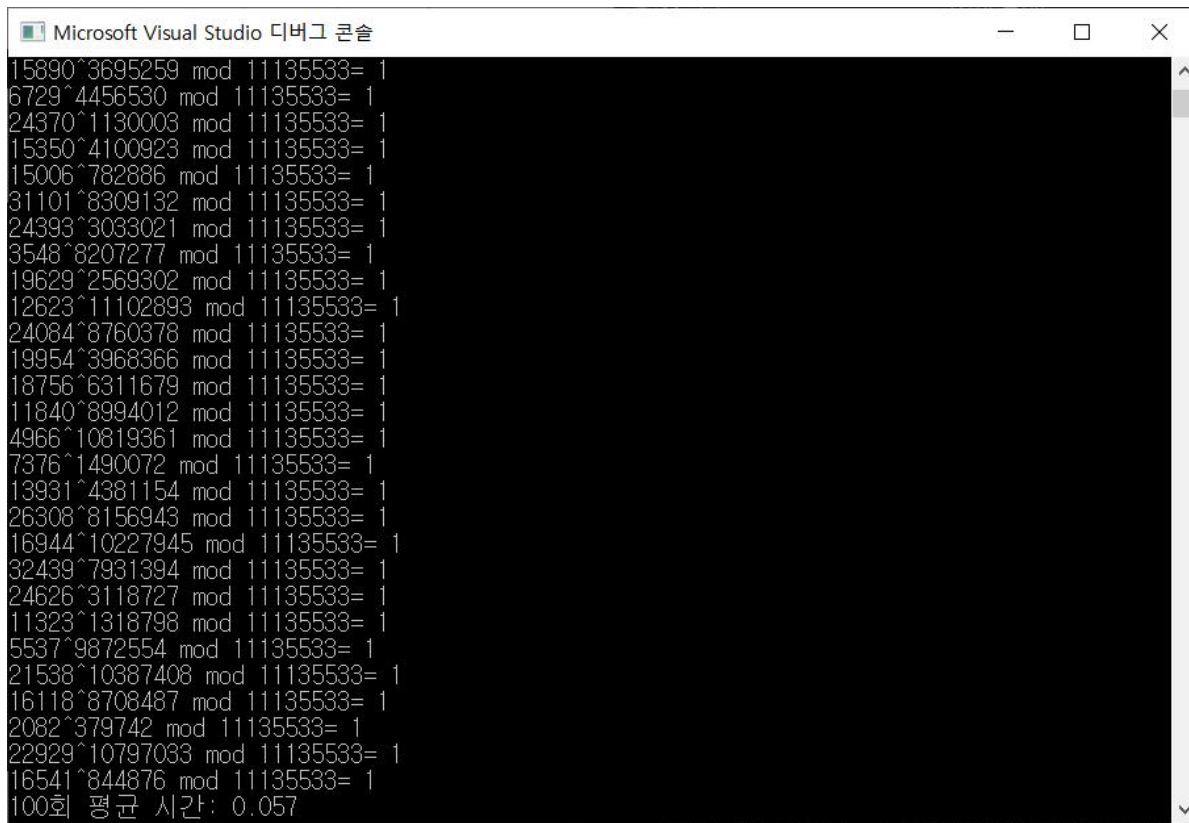

$N = 111547$

```
Microsoft Visual Studio 디버깅 콘솔
15890^75303 mod 111547= 1
6729^9681 mod 111547= 1
24370^30791 mod 111547= 1
15350^95320 mod 111547= 1
15006^94502 mod 111547= 1
31101^56550 mod 111547= 1
24393^59521 mod 111547= 1
3548^54736 mod 111547= 1
19629^35182 mod 111547= 1
12623^109002 mod 111547= 1
24084^111459 mod 111547= 1
19954^77866 mod 111547= 1
18756^98802 mod 111547= 1
11840^6755 mod 111547= 1
4966^71497 mod 111547= 1
7376^71577 mod 111547= 1
13931^59717 mod 111547= 1
26308^66098 mod 111547= 1
16944^102258 mod 111547= 1
32439^109893 mod 111547= 1
24626^44848 mod 111547= 1
11323^20215 mod 111547= 1
5537^38841 mod 111547= 1
21538^33172 mod 111547= 1
16118^8547 mod 111547= 1
2082^28235 mod 111547= 1
22929^29345 mod 111547= 1
16541^64692 mod 111547= 1
100회 평균 시간: 0.002
```

$N = 1022117$

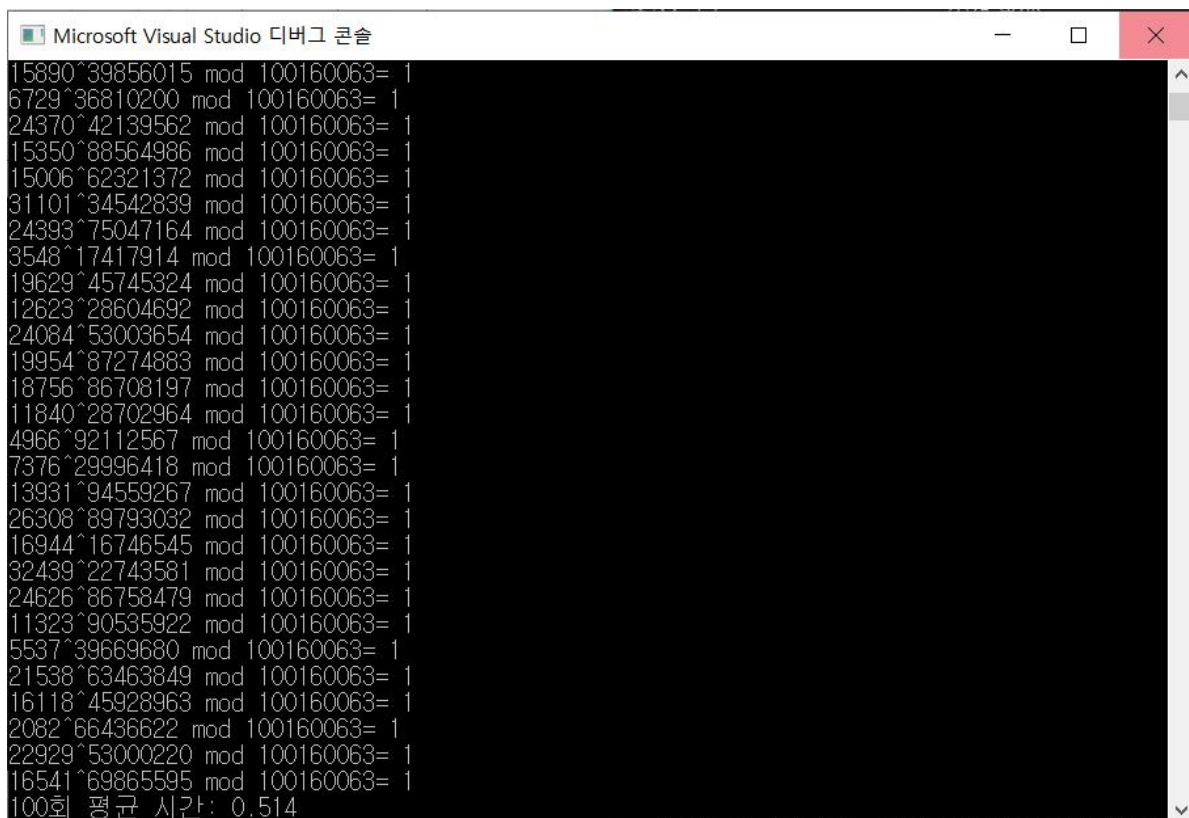
```
Microsoft Visual Studio 디버깅 콘솔
15890^990405 mod 1022117= 1
6729^62126 mod 1022117= 1
24370^907910 mod 1022117= 1
15350^329408 mod 1022117= 1
15006^855851 mod 1022117= 1
31101^367293 mod 1022117= 1
24393^1013988 mod 1022117= 1
3548^606990 mod 1022117= 1
19629^821901 mod 1022117= 1
12623^85669 mod 1022117= 1
24084^996441 mod 1022117= 1
19954^374906 mod 1022117= 1
18756^18038 mod 1022117= 1
11840^924308 mod 1022117= 1
4966^295356 mod 1022117= 1
7376^144532 mod 1022117= 1
13931^369858 mod 1022117= 1
26308^915313 mod 1022117= 1
16944^67743 mod 1022117= 1
32439^740459 mod 1022117= 1
24626^692521 mod 1022117= 1
11323^576639 mod 1022117= 1
5537^449126 mod 1022117= 1
21538^752422 mod 1022117= 1
16118^682278 mod 1022117= 1
2082^648519 mod 1022117= 1
22929^46316 mod 1022117= 1
16541^362292 mod 1022117= 1
100회 평균 시간: 0.007
```

N = 11135533



```
Microsoft Visual Studio 디버그 콘솔
15890^3695259 mod 11135533= 1
6729^4456530 mod 11135533= 1
24370^1130003 mod 11135533= 1
15350^4100923 mod 11135533= 1
15006^782886 mod 11135533= 1
31101^8309132 mod 11135533= 1
24393^3033021 mod 11135533= 1
3548^8207277 mod 11135533= 1
19629^2569302 mod 11135533= 1
12623^11102893 mod 11135533= 1
24084^8760378 mod 11135533= 1
19954^3968366 mod 11135533= 1
18756^6311679 mod 11135533= 1
11840^8994012 mod 11135533= 1
4966^10819361 mod 11135533= 1
7376^1490072 mod 11135533= 1
13931^4381154 mod 11135533= 1
26308^8156943 mod 11135533= 1
16944^10227945 mod 11135533= 1
32439^7931394 mod 11135533= 1
24626^3118727 mod 11135533= 1
11323^1318798 mod 11135533= 1
5537^9872554 mod 11135533= 1
21538^10387408 mod 11135533= 1
16118^8708487 mod 11135533= 1
2082^379742 mod 11135533= 1
22929^10797033 mod 11135533= 1
16541^844876 mod 11135533= 1
100회 평균 시간: 0.057
```

N = 100160063



```
Microsoft Visual Studio 디버그 콘솔
15890^39856015 mod 100160063= 1
6729^36810200 mod 100160063= 1
24370^42139562 mod 100160063= 1
15350^88564986 mod 100160063= 1
15006^62321372 mod 100160063= 1
31101^34542839 mod 100160063= 1
24393^75047164 mod 100160063= 1
3548^17417914 mod 100160063= 1
19629^45745324 mod 100160063= 1
12623^28604692 mod 100160063= 1
24084^53003654 mod 100160063= 1
19954^87274883 mod 100160063= 1
18756^86708197 mod 100160063= 1
11840^28702964 mod 100160063= 1
4966^92112567 mod 100160063= 1
7376^29996418 mod 100160063= 1
13931^94559267 mod 100160063= 1
26308^89793032 mod 100160063= 1
16944^16746545 mod 100160063= 1
32439^22743581 mod 100160063= 1
24626^86758479 mod 100160063= 1
11323^90535922 mod 100160063= 1
5537^39669680 mod 100160063= 1
21538^63463849 mod 100160063= 1
16118^45928963 mod 100160063= 1
2082^66436622 mod 100160063= 1
22929^53000220 mod 100160063= 1
16541^69865595 mod 100160063= 1
100회 평균 시간: 0.514
```

N = 1111355533

```
Microsoft Visual Studio 디버그 콘솔
15890^691222576 mod 1111355533= 1
6729^19653932 mod 1111355533= 1
24370^320728907 mod 1111355533= 1
15350^959530611 mod 1111355533= 1
15006^998857262 mod 1111355533= 1
31101^420943641 mod 1111355533= 1
24393^138959307 mod 1111355533= 1
3548^550979251 mod 1111355533= 1
19629^397402032 mod 1111355533= 1
12623^753640447 mod 1111355533= 1
24084^99626997 mod 1111355533= 1
19954^574948540 mod 1111355533= 1
18756^712402835 mod 1111355533= 1
11840^41581968 mod 1111355533= 1
4966^1098151752 mod 1111355533= 1
7376^141480863 mod 1111355533= 1
13931^587548166 mod 1111355533= 1
26308^1053016549 mod 1111355533= 1
16944^598901521 mod 1111355533= 1
32439^543978025 mod 1111355533= 1
24626^878172148 mod 1111355533= 1
11323^1091529178 mod 1111355533= 1
5537^532294541 mod 1111355533= 1
21538^148968494 mod 1111355533= 1
16118^795214255 mod 1111355533= 1
2082^515109553 mod 1111355533= 1
22929^991199819 mod 1111355533= 1
16541^415221401 mod 1111355533= 1
100회 평균 시간: 5.611
```

<시간 분석표>

두 자리 (21)	0.01
세 자리 (667)	0.02
네 자리 (1147)	0.02
다섯 자리 (10403)	0.02
여섯 자리 (111547)	0.02
일곱 자리 (1022117)	0.07
여덟 자리 (11135533)	0.057
아홉 자리 (100160063)	0.514
열 자리 (1111355533)	5.611

분석

이산대수 문제는 입력하는 숫자의 자리 수가 많아질수록 구하는데 걸리는 시간이 많이 소요되는 것을 확인할 수 있었다. 일곱자리까지는 나름 괜찮았지만 여덟 자리가 넘어가면서부터 시간이 이전보다 훨씬 많이 걸리는 것을 확인할 수 있었다. 이를 통해 공개키를 통한 암호화 알고리즘을 이용할 때 N 의 크기를 즉, 비트 수를 많게 할수록 암호키를 구하는 데 걸리는 시간이 기하 급수적으로 증가할 것으로 예상된다. 그러므로 키를 생성할 때는 이러한 테스트를 거쳐 최소한의 자리 수를 실험했을 때 너무 오래 걸려 구할 수 없을 정도의 자리 수로 만들어 주는 것이 좋을 것으로 예상된다.

소감

이번 과제를 수행하면서 공개키 암호의 알고리즘의 핵심인 이산대수 알고리즘에 대해서 직접 구현하고 조사해보고 분석해보면서 어느 장점이 있고 어떻게 구현되고 어느 단점이 있어 어떻게 보완되어야 하는지를 알 수 있었습니다. 그래서 수업시간에 배웠던 공개키 알고리즘에 대해서 더 자세하게 알 수 있게 된 것 같아서 뿌듯했습니다. 또한 대칭키암호와 접목해서 서로의 단점을 보완할 수 있을 것 같다는 것을 알 수 있게 되었습니다. 이번 과제를 통해서 이산대수 문제에 대해서도 지식을 얻을 수 있어서 매우 좋았고 코드를 구현하면서 프로그래밍 언어 실력에도 도움이 된 것 같아서 너무 뿌듯하고 좋았습니다.