Math-RSA

32기 안현진

문제 파일을 다운받고 압축을 풀면 텍스트 문서 3개가 있는걸 확인할 수 있다.

문제-텍스트 문서
수학 문제를 풀어서 RSA 암호를 해독해주세요!

RSA 암호라는 것을 처음 들어봐서 간단히 정리 해봤다.

1. RSA 암호화
RSA는 가장 대표적으로 사용되는 공개키 알고리즘이다.

#공개키 알고리즘: 발신자와 수신자가 서로 다른 키를 사용하여 데이터를 암호화하고 복호화하는 것으로, 비대칭 알고리즘이라고도 한다. 암호화를 위한 공개키(알려진 키)를 사용하여 메시지를 보내고, 특정 수신자만이 개인키(비밀키)를 사용하여 메시지를 복호화한다.

RAS 암호화를 사용하면 사용자는 공객적으로 공유할 수 있는 공개키라는 코드로 메시지를 암호화할 수 있다. 특정 RSA 알고리즘의 수학적 특성으로 인해, 사용자가 공개키를 사용하여 메시지를 암호화하고 나면 개인키로만 이를 복호화할 수 있다. 사용자는 공개키와 개인키 한 쌍을 가지고 있으며, 개인키는 비밀로 유지된다. RSA는 사용자가 사전에 키를 안전하게 배포하지 않은 경우 통신하는데 유용하다. 공개키와 비밀키 생성 과정에서 소인수분해가 사용된다. 암호화 과정 자체는 직접적으로 사용되지 않는다.

2. 문제 풀이

hint-텍스트 문서
3으로 나누었을 때 2가 남고, 5로 나누었을 때 3이 남고, 7로 나누었을 때 2가 남는 정수는?

일단 힌트 조건에 만족하는 정수를 구해보기로 했다. 간단한 파이썬 코드를 짜서 정수를 구했다.

```
a = 1
while a <= 1000:
    if a % 3 == 2:
        if a % 5 == 3:
            if a % 7 == 2:
                print(a)
    a += 1
-> 23, 128, 233, 338, 443, 548, 653, 758, 863, 968
```

코드를 통해 힌트에 만족하는 1000이하의 정수를 구했다.

보통 해커들이 쉽게 풀지 못하도록 소인수분해을 이용한 키 생성 과정에 1024bit 이상의 숫자를 사용한다고 한다. 1 ~ 2^(1024-1) 범위의 숫자이니 사실상 최대값은 무한대에 가까운 정수인 것이다. 하지만 ctf문제로 주어진 상황이니 이 그나마 작은 값의 정수를 주지 않았을까...?하는 마음으로 최대값을 1000으로 잡았다.

hint에 부합하는 정수들을 찾아보았으니 다음 힌트로 넘어가보기로 했다.

rsa-텍스트 문서

? = 3

text1_1 =

14789627007255136019575345436328229942606248517474575935121184648992891024175322481
97352857448458376380839443503589087859095842621324159214616930278992361860753830108
52224067091477810924118719861660629389172820727449033189259975221664580227157731435
89416391784198080202106884054985329916643725718107237276169 3

text1_2 =

15533331568434192825972570642762409799236210100495627939140207857266587784857774931
84652870922291113697752126197841006298022719778433479467190657576004691360598242468
19070794143738111977731658922334606275707854421245356041436150606001398623546617591
70185390948862797714097748009536946058226465839144178144877

text2_1 =

95979365485314068430194308015982074476106529222534317931594712046922760584774363858
26799569833941733598654334729270749583318292143939898354042500410599058381311306512
48367954707603248766492255769216552333466304226695517136024239877938224592967614034
56611062240111812805323779302474406733327110287422659815403

text2_2 =

60485293124433437141118039812909015910813570340644532997460020712280153885886509927
20459600638557998501631794938590261123892492386960431506339288443362603044569373919
06340535555792838073582565505569468879422201808926791767065970489261814589286345877
01939862436245447175330265870303082187081443518060917411902

text3_1 =

95649308318281674792416471616635514342255502211688462925255401503618542159533496090
63894778481845634789683316850817942585327774029024229744546511810651365722908240687
77323153193404030489311235304355013718817408593357938041943156759721926490010743789
34213623075830325229416830786633930007188095897620439987817

text3_2 =

85737153723532791682986185847254831480032106939823543174113563634075907849864317401
40816288671037551112705276005588035122632359847630917632156269530986698051761172466
52842708176111589475321254513301375663040905620724876412220108649113934289628011988
70096341090040999154522110372919507107408751138245379355338

와! 이게 뭘까 1-1, 1-2, 2-1...이렇게 쌍으로 존재하는걸 보면 공개키와 비밀키를 의미하는걸까?

그래서 공개키와 비밀키를 구하는 공식을 찾아봤다.

1. 공개키 생성

- 공개키는 n과 e로 구성됩니다.
  - n은 두 개의 큰 소수 p와 q의 곱입니다: $n = p \times q$.
  - e는 $\phi(n)$과 서로 소인 값으로 선택됩니다. 여기서 $\phi(n)$은 p와 q로부터 계산된 오일러의 피 함수 값입니다.
- n을 생성하기 위해서는 두 개의 큰 소수 p와 q를 곱하는 과정이 필요합니다.

2. 비밀키 생성

- 비밀키는 d로 구성되며, 이는 공개키의 e와 n을 사용하여 계산됩니다.
  - d는 $e \times d \equiv 1 \pmod{\phi(n)}$를 만족하는 값입니다. 여기서 $\phi(n) = (p-1) \times (q-1)$입니다.
  - $\phi(n)$을 계산하기 위해서는 p와 q 값을 알아야 합니다.

근데 오일러 공식이 쓰인다. 음.. 난이도 하인 문제에서 이런 계산을 요구할까? 싶어서 일단 이 방법은 보류했다.

그렇다면 rsa 텍스트 문서의 내용이 의미하는 바는 뭘까? '?=3' 그리고 의미없어보이는 숫자들의 나열. rsa 관련 ctf 블로그들을 살펴봤는데 보통 e,n 등의 값을 알려준다. 하지만 이 문제에서는 직접적으로 값을 알려주지는 않고 있다. text1_1...이런게 변수인가?

e값이 작은 경우에는 특별한 툴 없이 파이썬 코드로도 해독이 가능하다고 해서 이 방법을 시도해보려고 한다.

```
c = 14789627007255136019575345436328229942606248517474575935121184648992891024175322481973528574484583763808394435035890878590958426213241592146169302789923618607538301085222406709147781092411871986166062938917282072744903318925997522166458022715773143589416391784198080202106884054985329916643725718107237276169
3

# c의 세제곱근을 계산합니다.
m = int(c ** (1/3))

# 계산된 m을 16진수로 변환한 후, 이를 바이트로 변환합니다.
result = bytes.fromhex('%x' % m)

print(result)
->b'%\xc73\xf2\xf7\x08\xfa\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00'
```

결과 값을 아스키코드로 변환해봤다. 하지만 의미있는 문자열은 보이지 않는다. 나중에 더 공부한 뒤에 다시 도전해야 할 것 같다.

<참고자료>

RSA, 제대로 이해하기 (1) (tistory.com)
RSA 암호화란 무엇입니까? | Veritas
[RSA] RSA 공격법 (tistory.com)
보안맨 (tistory.com)