Практическая работа

Everything is Big (70 Pts)

Чубань Артем, Конрат Климентий

Балтийский федеральный университет имени Иммануила Канта

2 июля 2021 г.

Формулировка задачи

Даны следующие данные:

- c - Зашифрованное сообщение
- n - Модуль
- e - Публичная экспонента

Задача:

- Найти исходное сообщение

Ссылка на задачу

Выбор атаки

Наше решение данной задачи, главным образом, является реализацией атаки Винера, основанное на том факте, что приватный ключ d слишком мал по отношению к модулю n.

С полным кодом вы можете ознакомиться по ссылке ниже
Код на GitHub

Метод решения

Перед атакой такого типа, нужно понимать следующее:

- Мы знаем, что $e \cdot d = 1 \ (mod \ \varphi)$, $\varphi = lcm(p-1, q-1)$
- Значит существует такое целок K, что $e \cdot d = K \cdot \varphi + 1$
- Пусть $G = gcd(p-1, q-1)$, тогда $e \cdot d = \frac{K}{G} \cdot \varphi + 1$
- Обозначим $k = \frac{K}{gcd(K,G)}$, $g = \frac{G}{gcd(K,G)}$
- Тогда получим $e \cdot d = \frac{k}{g} \cdot \varphi + 1$
- Или $e \cdot d \cdot g = k \cdot (p-1)(q-1) + g$

isPerfectSqr(num)

```python
def isPerfectSqr(num):
    x = num // 2
    seen = set([x])

    while x * x != num:
        x = (x + (num // x)) // 2
        if x in seen:
            return False
        seen.add(x)

    return True
```

Функция возвращает Истину, если аргумент является идеальным квадратом.

Алгоритм работы:

1. $x = \frac{1}{2}num$
2. Цикл while работает, пока $x^2 \neq num$ или x не повторилось
3. Следующий $x = \frac{1}{2}(x + \frac{num}{x})$

Такой алгоритм дает преимущество по скорости по сравнению с обычным перебором

rational_to_contfrac(e, n)

```
def rational_to_contfrac(e, n):
  while e:
        a = e // n
        yield a
        e, n = n, e - a * n
```

Функция раскладывает дробь в цепную, где *e* - числитель, *n* - знаменатель. Возвращаемое значение - список-итератор

Алгоритм работы:

1. Цикл while работает пока $e \neq 0$
2. Вычисляем *a* - целая часть от деления аргументов
3. Следуя алгоритму Евклида, меняем переменные местами, вычитая целую часть

contfrac_to_rational_iter(contfrac)

```
def contfrac_to_rational_iter(contfrac):
  n0, d0 = 0, 1
  n1, d1 = 1, 0

  for q in contfrac:
      n = q * n1 + n0
      d = q * d1 + d0
      yield n, d

      n0, d0 = n1, d1
      n1, d1 = n, d
```

Функция вычисляет «подходящие дроби» используя циклическую дробь, вычисленную предыдущей функцией. Для этого используются рекурентные соотношения теории чисел, которые можно видеть внутри цикла «for» на слайде.

Возвращает итератор-кортеж, где первая компонента – числитель, вторая – знаменатель.

convergents_from_contfrac(contfrac)

```
def convergents_from_contfrac(contfrac):
    nn, dd = 1, 0

    for i, (n, d) in enumerate(contfrac_to_rational_iter(contfrac)):
        if i % 2 == 0:
            yield n + nn, d + dd
        else:
            yield n, d
        nn, dd = n, d
```

Преобразует цепные дроби в сходящиеся числовые ряды.

Этот алгоритм очень похож на алгоритм Евклида и называется Алгоритм Непрерывной Дроби. Его особенность заключается в рассмотрении честности и нечетности.

get_private_exponent(e, n)

```python
def get_private_exponent(e, n):
  for k, dg in convergents_from_contfrac(rational_to_contfrac(e, n)):
    edg = e * dg
    phi = edg // k
    x = n - phi + 1

    if x % 2 == 0 and isPerfectSqr((x//2)**2 - n):
      g = edg - phi * k
      return dg // g

  return 0
```

Функция возвращает приватный ключ d, если атака успешна, или 0 в противном случае.

Принцип работы:

1. Каждую итерацию цикла for вычисляем $e \cdot d \cdot g$ и φ согласно методу решения
2. Следуя алгоритму атаки Винера, проверяем, что x - четное и $(\frac{x}{2})^2 - n$ - квадрат целого числа
3. Если оба условия верны, то получаем приватную экспоненту по формуле из $\frac{d \cdot g}{g}$, где $d \cdot g$ нам известно, а $g = e \cdot d \cdot g - \varphi \cdot k$

__main__

```python
if __name__ == "__main__":
    from Crypto.Util.number import long_to_bytes
    from parametrs import N, E, C

    D = get_private_exponent(E, N)
    plainMessage = long_to_bytes(pow(C, D, N)).decode()

    print(plainMessage)
```

Основное тело программы. Реализует атаку Винера, а затем расшифровывает исходное сообщение.

Принцип работы:

1. Импортируем long_to_bytes, чтобы перевести число в бинарную строку, а также параметры из задачи
2. Получаем приватную экспоненту
3. Расшифровываем $plainMessage = c^d \ (mod\ n)$ и выводим его на экран

Итог

Таким образом, при исходных данных

n =
0x8da7d2ec7bf9b322a539afb9962d4d2ebeb3e3d449d709b80a51dc680a14c87ffa863edfc7b5a2a542a0fa610febe
2d967b58ae714c46a6eccb44cd5c90d1cf5e271224aa3367e5a13305f2744e2e56059b17bf520c95d521d34fdad3b0
c12e7821a3169aa900c711e6923ca1a26c71fc5ac8a9ff8c878164e2434c724b68b508a030f86211c1307b6f90c0cd48
9a27fdc5e6190f6193447e0441a49edde165cf6074994ea260a21ea1fc7e2dfb038df437f02b9ddb7b5244a9620c8ec
a858865e83bab3413135e76a54ee718f4e431c29d3cb6e353a75d74f831bed2cc7bdce553f25b617b3bdd9ef901e24
9e43545c91b0cd8798b27804d61926e317a2b745

e =
0x86d357db4e1b60a2e9f9f25e2db15204c820b6e8d8d04d29db168c890bc8a6c1e31b9316c9680174e128515a002
56b775a1a8ccca9c6936f1b4c2298c03032cda4dd8eca1145828d31466bf56bfcf0c6a8b4a1b2fb27de7a57fae743004
8d7590734b2f05b6443ad60d89606802409d2fa4c6767ad42bffae01a8ef1364418362e133fa7b2770af64a68ad50ad
8d2bd5cebb99ceb13368fb31a6e7503e753f8638e21a96af1b6498c18578ba89b98d70fa482ad137d28fe701b4b77b
aa25d5e84c81b26ee9bddf8cbb51a071c60dd57714de379cd4bc14932809ba18524a0a18e4133665cfc46e2c4fcfbc2
8e0a0957e5513a7307c422b87a6182d0b6a074b4d

c =
0x6a2f2e401a54eeb5dab1e6d5d80e92a6ca189049e22844c825012b8f0578f95b269b19644c7c8af3d544840d380e
d75fdf86844aa8976622fa0501eaec0e5a1a5ab09d3d1037e55501c4e270060470c9f4019ced6c4e67673843daf2fd7
1c64f3dd8939ae322f2b79d283b3382052d076ebe9bb50b0042f1f7dd7beadf0f5686926ade9fc8370283ead781a21
896e7a878d99e77c3bb1f470401062c0e0327fd85da1cf12901635f1df310e8f8c7d87aff5a01dbbecd739cd8f364620
60d0eb237af8d613e2d9cebb67d612bcfc353ef2cd44b7ac85e471287eb04ae9b388b66ea8eb32429ae96dba5da82
06894fa8c58a7440a127fceb5717a2eaa3c29f25f7

ответом на задачу является флаг
crypto{s0m3th1ng5_c4n_b3_t00_b1g}

Библиография


Michael J. Wiener
Cryptanalysis of Short RSA Secret Exponents — 1989 August 3