

# d-phi-enc (HackTM CTF)

Кулигин Данил и Борзенко Михаил

БФУ им. Иммануила Канта ИФМНиИТ  
Компьютерная безопасность, 3 курс

1 июля 2023 г.

# Условие задачи d-phi-enc

Описание с CryptoHack

Описание к заданию: *"В CTF есть много людей, которые ошибочно шифруют  $p$ ,  $q$  в RSA. Но на этот раз..."*

Так же к задаче прилагаются файлы `chall.py` и `output.txt`. `output.txt` содержит  $n$  - произведение  $p$  и  $q$ , `enc_d` - зашифрованная секретная экспонента, `enc_phi` - зашифрованная функция Эйлера, `enc_flag` - зашифрованное сообщение. В `chall.py` содержится код для генерации этих значений и открытая экспонента  $e$

Понятно, что нам нужно каким-то образом извлечь  $d$  или  $\phi$  на основе `enc_d` и `enc_phi`.

# Анализ задачи d-phi-enc

## Предварительный анализ

Для начала проведем анализ данных нам чисел и проверим криптосистему на уязвимость к изученным нами атакам, в числе которых: Атака повторным шифрованием, Атака Винера, Атака встреча посередине, Атака методом Ферма.

Нам даны:

$n = 2447638356...$

$enc\_d = 2385197103...$

$enc\_phi = 3988439673...$

$enc\_flag = 2403368891...$

$e = 3$

Упрощая, нам дано  $n$  длиной 2048 бит. Также мы знаем, что длина  $p$  и  $q$  равна 1024 бита. Очевидно, что разложить  $n$  на множители вряд ли будет эффективным решением.

# Анализ задачи d-phi-enc

## Предварительный анализ

- Атака “встреча посередине” не будет успешной, т. к. обычно она срабатывает при малой длине сообщения ( $l < 64$  бит) и наличии разложения на два примерно равных множителя, битовая длина которых меньше  $l/2$ .
- Атака методом Ферма также не принесёт результатов, т. к. данная атака могла бы быть успешна, если  $p$  и  $q$  “близкие” друг к другу числа (половина старших цифр числа равны).
- Атака Винера гарантированно успешна, при  $d < d_{\text{кр}}$ , где  $d_{\text{кр}} = n^{1/4}/\sqrt{2a}$ ,  $a = (h+1)/\sqrt{h}$ ,  $h = p/q$ .

# Анализ задачи d-phi-enc

## Предварительный анализ

- Как мы можем заметить, для вычисления  $d_{кр}$  требуется знать  $p$  и  $q$ , а мы их пока что не знаем, но мы знаем точно, что попытка атаки Винера (метод которой заключается в разложении дроби  $e/n$  в цепную дробь для последующего нахождения  $d$ ) не принесла результата.
- Атака повторным шифрованием также не дала быстрого результата, из чего можем сделать вывод, что порядок  $e$  по модулю сообщения достаточно велик.

# Решение задачи d-phi-enc

## Теоретические расчёты

- Из алгоритма работы RSA мы знаем, что  $d$  это обратное число  $e$  по модулю  $\varphi(n)$ .

$$\begin{aligned}e * d &\equiv 1 \pmod{\varphi(n)} \\e * d &= k_1 * \varphi(n) + 1\end{aligned}$$

- Т. к.  $d < n$  и  $e = 3$  можем сделать вывод, что  $0 < k_1 < 3$ .  
Далее распишем  $d_{enc}$  используя сравнение выше.

$$\begin{aligned}d_{enc} &\equiv d^3 \pmod{n} \\e^3 * d_{enc} &\equiv e^3 * d^3 \pmod{n} \\27d_{enc} &\equiv (e * d)^3 \pmod{n} \\27d_{enc} &\equiv (k_1 * \varphi(n) + 1) \pmod{n} \\27d_{enc} &\equiv (k_1^3 * \varphi^3(n) + 3k_1^2 * \varphi^2(n) + 3k_1 * \varphi(n) + 1) \pmod{n}\end{aligned}$$

# Решение задачи d-phi-enc

## Теоретические расчёты

- Заменяем  $\varphi^3(n)$  на  $\varphi_{enc}$ , т. к.  $e = 3$ , а сравнение выполняется по модулю  $n$ , также как и шифрование.

$$27 * d_{enc} \equiv (k1^3 * \varphi_{enc} + 3k1^2 * \varphi^2(n) + 3k1 * \varphi(n) + 1) \pmod{n}$$

- Перенеся всё в одну сторону можно получить квадратное сравнение от  $\varphi(n)$

$$3k1^2 * 2(n) + 3k1 * (n) + k1^3 * \varphi_{enc} - 27d_{enc} + 1 \equiv 0 \pmod{n}$$

# Решение задачи d-phi-enc

## Теоретические расчёты

- Все переменные кроме  $\varphi(n)$  даны. Однако, т.к.  $n$  не является простым числом, то решить такое сравнение можно только через систему сравнений по модулям множителей  $n$ , но в нашем случае множители неизвестны. Поэтому попробуем расписать  $\varphi(n)$  для упрощения решения.

$$\begin{aligned}\varphi(n) &= (p-1) * (q-1) = pq - p - q + 1 = n - p - q + 1 \\ \varphi(n) &= (p+q) + 1 \pmod{n}\end{aligned}$$



# Решение задачи d-phi-enc

## Теоретические расчёты

- Обозначим  $x = p + q$ , тогда

$$3k1^2 * \varphi^2(n) + 3k1 * \varphi(n) + k1^3 * \varphi_{enc} - 27d_{enc} + 1 \equiv 0 \pmod{n}$$

$$3k1^2 * (1 - x)^2 + 3k1 * (1 - x) + k1^3 * \varphi_{enc} - 27d_{enc} + 1 \equiv 0 \pmod{n}$$

$$3k1^2 * (1 - 2 * x + x^2) + 3k1 * (1 - x) + k1^3 * \varphi_{enc} - 27d_{enc} + 1 \equiv 0 \pmod{n}$$

$$3k1^2 * x^2 + (-6k1^2 - 3k1) * x + (3k1^2 + 3k1 + k1^3 * \varphi_{enc} - 27d_{enc} + 1) \equiv 0 \pmod{n} \quad (1)$$

# Решение задачи d-phi-enc

## Теоретические расчёты

- Далее зная что  $x = p + q$ , можем сделать вывод что  $x$  гораздо меньше  $n$ . Допустим  $p > q$ .

$$x^2 = (p + q)^2 = p^2 + q^2 + 2 * p * q = p^2 + q^2 + 2n$$

- Зная что  $p, q$  сгенерированы длиной 1024 бит,  $p/q < 2$ .

Можем записать такое неравенство:

$$p \leq 2 * q, \text{ следовательно } p^2 \geq 4 * q^2. \text{ А также } q^2 \leq n$$

$$p^2 + q^2 + 2 * n \leq 4 * q^2 + q^2 + 2 * n$$

$$x^2 \leq 5 * q^2 + 2 * n$$

$$x^2 \leq 5 * n + 2 * n$$

$$x^2 \leq 7 * n$$

# Решение задачи d-phi-enc

## Теоретические расчёты

- Используя коэффициент  $a$  из квадратного сравнения (1) распишем неравенство для  $x^2$ .

$$3k_1^2 * x^2 \leq 3 * 2^2 * x^2, \text{ так как } k_1 \leq 2.$$

$$3k_1^2 * x^2 \leq 12 * 7 * n$$

$$3k_1^2 * x^2 \leq 84 * n$$

$$3k_1^2 * x^2 + (-6k_1^2 - 3k_1) * x + (3k_1^2 + 3k_1 + k_1^3 * \varphi_{enc} - 27d_{enc} + 1) < 84n$$

- Мы знаем все числа из  $\mathbf{b}$  и  $\mathbf{c}$  коэффициентов неравенства выше, делаем вывод что  $\mathbf{b}$  и  $\mathbf{c}$  отрицательны, поэтому меняем знак на строгий.

$$3k_1^2 * x^2 + (-6k_1^2 - 3k_1) * x + (3k_1^2 + 3k_1 + k_1^3 * \varphi_{enc} - 27d_{enc} + 1) = k_2 * n, \text{ где } k_2 \leq 84, k_2 \in \mathbb{Z}.$$

# Решение задачи d-phi-enc

## Теоретические расчёты

- Таким образом мы можем перебрать все возможные  $k^2$  чтобы решить уравнение, корнем которого будет  $x$ ,  
 $x = p + q$ .
- Для решения таска нам нужно знать  $\varphi(n)$ , оно находится с помощью выражения ниже, но мы также можем подсчитать  $p$  и  $q$  чтобы проверить наше решение  $p * q = n$ .  
 $\varphi(n) = n - p - q + 1$ , домножим на  $p$  и перенесем всё в одну сторону.

$$pn - p^2 - pq + p - p * \varphi(n) = 0$$

$$p^2 - pn - p + p * \varphi(n) + n = 0$$

$$p^2 - p * (n - \varphi(n) + 1) + n = 0$$

# Решение задачи d-phi-enc

## Теоретические расчёты

- Решением данного квадратного уравнения будут  $p$  и  $q$ .  
Далее мы можем проверить выражение  $p * q = n$  и переходить к финальному шагу.  
Вычисляем  $d = e^{-1} \pmod{\varphi(n)}$   
Вычисляем флаг:  $\text{flag} = \text{enc\_flag}^d \pmod{n}$
- С помощью функции `long_to_bytes` вычисляем текстовое значение флага, которое равно:  
`b"HackTM{Have you warmed up? If not, I suggest you consider the case where e=65537, although I don't know if it's solvable. Why did I say that? Because I have to make this flag much longer to avoid solving it just by calculating the cubic root of enc_flag.}"`

# Решение задачи d-phi-enc

Код для решения данной задачи

```
from Crypto.Util.number import long_to_bytes

n = 2447638356...
enc_d = 2385197103...
enc_phi = 3988439673...
enc_flag = 2403368891
e = 3

for k1 in range(1, 3):
    a = 3*(k1^2)
    b = -(6*(k1^2)+3*k1)
    c = 3*(k1^2) + 3*k1 + (k1^3)*enc_phi - 27*int(enc_d) + 1

    #f = a*(x^2) + b*x + c
    det = b^2 - 4*a*c
    for k2 in range(85):
        c -= n
        det = b^2 - 4*a*c
        if (is_square(det)): break
    if (is_square(det)): break
```

# Решение задачи d-phi-enc

Код для решения данной задачи

```
qrs_det = sqrt(b^2 - 4*a*c)
print("qrs_det=", qrs_det)

cand_x = (-1*b + qrs_det)/(2*a) #x=p+q
phi = n - cand_x + 1
print("phi=", phi)

p = ((n + 1 - phi) + sqrt(((n + 1 - phi) ^ 2) - 4 * n))/ 2
q = n // p
print("p:␣", type(p), p)
print("q:␣", type(q), q)
assert(p*q == n)

d = pow(e, -1, phi)
print("flag=", (pow(enc_flag, d, n)))
```

# Концепция RSA

## Математическая модель

- Асимметричные криптографические системы основаны на так называемых односторонних функциях с секретом. Например, операция возведения числа в степень по модулю:

$$\begin{aligned}c &\equiv f(m) \equiv m^e \pmod{n} \\m &\equiv f^{-1}(c) \equiv c^d \pmod{n} \\d &\equiv e^{-1} \pmod{\varphi(n)}\end{aligned}$$



# Концепция RSA

## Математическая модель

- Асимметричные криптографические системы основаны на так называемых односторонних функциях с секретом. Например, операция возведения числа в степень по модулю:

$$\begin{aligned}c &\equiv f(m) \equiv m^e \pmod{n} \\ m &\equiv f^{-1}(c) \equiv c^d \pmod{n} \\ d &\equiv e^{-1} \pmod{\varphi(n)}\end{aligned}$$

- ***c*** получено возведением в степень по модулю числа ***m***. Назовём это действие шифрованием  
***m*** - открытый текст, а ***c*** - шифртекст  
Пара чисел (***e***, ***n***) - открытый ключ  
***d*** - закрытый ключ

# Концепция RSA

## Математическая модель

- Пусть  $n = p \cdot q$ , где  $p$  и  $q$  – некоторые разные простые числа. Для такого  $n$  функция Эйлера имеет вид:

$$\varphi(n) = (p - 1) \cdot (q - 1)$$

# Концепция RSA

## Математическая модель

- Пусть  $n = p \cdot q$ , где  $p$  и  $q$  – некоторые разные простые числа. Для такого  $n$  функция Эйлера имеет вид:

$$\varphi(n) = (p - 1) \cdot (q - 1)$$

- Выберем такое число  $e$ :

$$e \in [3, \varphi(n) - 1]$$
$$\text{НОД}(e, \varphi(n) - 1) = 1$$

Вычислим  $d$ :

$$d \equiv e^{-1} \pmod{\varphi(n)}$$

# Концепция RSA

## Математическая модель

- Возьмём в качестве сообщения число  $m \in [1, n - 1]$   
Чтобы зашифровать его, необходимо возвести его в степень  $e$  по модулю  $n$

$$c \equiv m^e \pmod{n}$$

Отметим также, что  $c \in [1, n - 1]$ , как и  $m$ . Расшифруем шифртекст, возведя его в степень закрытого ключа  $d$ :

$$m' \equiv c^d \pmod{n}$$

# Используемая литература



Математика криптографии и теория шифрования. Лекция 13: Квадратичное сравнение.

<https://intuit.ru/studies/courses/552/408/lecture/9370>



RSA: от простых чисел до электронной подписи

<https://habr.com/ru/post/534014/>



Криптоанализ RSA. Сонг Ян, 2011 год. ISBN 978-5-93972-873-7.



The CrypTool Book: Learning and Experiencing Cryptography with CrypTool and SageMath. Prof. Bernhard Esslinger and the Development Team of the Open-Source Software CrypTool. Edition 12 (2018).

<https://www.cryptool.org>