Федеральное государственное автономное образовательное учреждение высшего образования
«Национальный исследовательский университет ИТМО»

Факультет программной инженерии и компьютерной техники

Лабораторная работа 5

«Атака на алгоритм шифрования RSA методом бесключевого чтения»

Вариант № 12

Группа: P34102

Выполнил: Лапин А.А.

Проверил:
Рыбаков С.Д.

Санкт-Петербург
2024г.

Оглавление

Введение

Цель работы: изучить атаку на алгоритм шифрования RSA посредством метода бесключевого чтения.

Текст задания

Вариант	Модуль, N	Экспоненты		Блок зашифрованного текста	
		e1	e2	C1	C2
12	385751370271	365797	1109663	58541562205 167003685579 381877628242 256218527098 164244249864 6588741823 180308234660	78032032470 13064174635 326727914830 364066420370 177576861402 65863828523 111437045566

Ход работы

Будем строить последовательность: $c_1 = c,\ c_i = c_{i-1}^e \mod N, i > 1$.

$N = 385751370271$

$e_1 = 365797$

$e_2 = 1109663$

$c_1 = 58541562205...$

$c_2 = 78032032470...$

$\Rightarrow$

$c_1 = m^{e_1} \mod N$

$c_2 = m^{e_2} \mod N$

Используя расширенный алгоритм Евклида, находим r и s, такие что $re_1 + se_2 = 1$

Тогда $m = c_1^r c_2^s \mod N$

Программная реализация

Listing 1: main.py

```python
import math
from omegaconf import DictConfig
import hydra
from tqdm import tqdm
import logging

logger = logging.getLogger(__name__)


def int_to_bytes(m):
    """
    Convert an integer to bytes.
    """
    hex_str = hex(m)[2:]
    if len(hex_str) % 2:
        hex_str = '0' + hex_str
    return bytes.fromhex(hex_str)


def extended_gcd(a, b):
    """
    Extended Euclidean Algorithm.
    Returns a tuple of (gcd, x, y), where gcd is the gcd of a and b,
    and x, y satisfy the equation: a*x + b*y = gcd
    """
    if a == 0:
        return (b, 0, 1)
    else:
        gcd, x1, y1 = extended_gcd(b % a, a)
        x = y1 - (b // a) * x1
        y = x1
        return (gcd, x, y)


def infinite_reading_attack(y1, e1, y2, e2, N):
```

```
    """
    Perform RSA cryptanalysis using the Infinite Reading Attack (Бесключевое чтение).
    Given two ciphertexts y1 = x^e1 mod N and y2 = x^e2 mod N,
    recover the original plaintext x.
    """
    gcd, r, s = extended_gcd(e1, e2)
    if gcd != 1:
        logger.error(
            "Exponents e1 and e2 are not coprime. Attack cannot be performed.")
        return None

    # Compute x = (y1^r * y2^s) mod N
    x = (pow(y1, r, N) * pow(y2, s, N)) % N

    logger.debug(f"Recovered plaintext x: {x}")
    return x


@hydra.main(version_base=None, config_path=".", config_name="config")
def main(cfg: DictConfig):
    N = cfg.N
    e1 = cfg.e1
    e2 = cfg.e2
    c1 = cfg.c1
    c2 = cfg.c2

    print(f"N = {N}")
    print(f"e1 = {e1}")
    print(f"e2 = {e2}")
    print(f"c1 = {c1}")
    print(f"c2 = {c2}")

    # Perform Infinite Reading Attack on each ciphertext pair
    print("Performing Infinite Reading Attack on ciphertext pairs...")
    decrypted_bytes = b''

    for idx, (y1, y2) in tqdm(enumerate(zip(c1, c2), start=1), total=len(c1), desc="
        Decrypting Ciphertext Pairs"):
        logger.debug(f"Decrypting ciphertext pair {idx}: y1={y1}, y2={y2}")
        plaintext_int = infinite_reading_attack(y1, e1, y2, e2, N)
        if plaintext_int is None:
            print(f"Failed to decrypt ciphertext pair {idx}.")
            continue
        decrypted_bytes += int_to_bytes(plaintext_int)

    try:
        plaintext = decrypted_bytes.decode('cp1251')
        print(f"Plaintext: {plaintext}")
    except UnicodeDecodeError:
        print("Decrypted bytes could not be decoded to cp1251. Raw bytes:")
        print(decrypted_bytes)


if __name__ == "__main__":
    main()
```

Listing 2: config.yaml

```yaml
# RSA Configuration Parameters

# The modulus N, which is the product of two primes p and q.
N: 385751370271

# The public exponent e.
e1: 365797
e2: 1109663

# The list of ciphertext blocks to be decrypted.
c1:
  - 58541562205
  - 167003685579
  - 381877628242
  - 256218527098
  - 164244249864
  - 6588741823
  - 180308234660
  - 174572441677
  - 259951955034
  - 378589342820
  - 319378579620
  - 21405495597
  - 226860843155
c2:
  - 78032032470
  - 13064174635
  - 326727914830
  - 364066420370
  - 177576861402
  - 65863828523
  - 111437045566
  - 124743274954
  - 119577259869
  - 85769669875
  - 4688914942
  - 261002397567
  - 341722428571
```

Результаты работы программы

Listing 3: Вывод в консоль

```
> python main.py
N = 385751370271
e1 = 365797
e2 = 1109663
c1 = [58541562205, 167003685579, 381877628242, 256218527098, 164244249864,
    6588741823, 180308234660, 174572441677, 259951955034, 378589342820,
    319378579620, 21405495597, 226860843155]
c2 = [78032032470, 13064174635, 326727914830, 364066420370, 177576861402,
    65863828523, 111437045566, 124743274954, 119577259869, 85769669875, 4688914942,
    261002397567, 341722428571]
Performing Infinite Reading Attack on ciphertext pairs...
```

```
Decrypting Ciphertext Pairs: ████████████████████████████████100%|| 13/13
    [00:00<00:00, 53667.28it/s]
Plaintext: из исходного пакета, в котором переданы эти 900 б.
```

Заключение

В ходе выполнения лабораторной работы была реализована атака на алгоритм шифрования RSA методом бесключевого чтения.