Федеральное государственное автономное образовательное учреждение высшего образования "Национальный исследовательский университет ИТМО"

Лабораторная работа №1

по дисципение "Информационная безопасность"

Атака на алгоритм шифрования RSA посредством метода Ферма

Вариант 10

Выполнил: студент Саржевский И.А.

Группа: P3402

Преподаватель: к.т.н., доцент
Маркина Т.А.

г. Санкт-Петербург

2021 г.

Лабораторная работа №1

Атака на алгоритм шифрования RSA посредством метода Ферма

Цель работы

Изучить атаку на алгоритм шифрования RSA посредством метода Ферма.

Задание

Алгоритм шифрования RSA состоит из следующих шагов:

1. Выбираются простые числа p и q, вычисляется n = p * q;
2. $\phi(n) = (p-1)(q-1)$;
3. Находится число e, взаимно простое $\phi(n)$;
4. Вычисляется d, такое, что de эквивалентно единице по модулю $\phi(n)$.

(n, e) - публичный ключ. Для шифрования сообщение разбивается на блоки $t(< n)$, зашифрованный текст: $c = t^e \ mod \ n$.

Для дешифрования используется приватный ключ (n, d): $t = c^d \ mod \ n$.

В данной лабораторной работе рассматривается атака с помощью метода Ферма. Он заключается в решении уравнения $t^2 - w^2 = n$, иными словами, поиске t, при котором $t^2 - n$ - простое число. Поиск начинается с $\sqrt{n}$, так как это минимальное значение, при котором $t^2 - n \geq 0$. Затем текущее значение инкрементируется на каждой итерации, производится поиск значения, при котором $\sqrt{t^2 - n}$ является целым числом w. В таком случае $p = t + w; q = t - w$, зная которые можно расшифровать текст.

Разработанная программа принимает путь к yaml-файлу, в котором описаны исходные данные: N, e и блоки данных C.

Исходные данные

```
N: 77027476849549
e: 2936957
C:
  - 18937689886043
  - 6667195679130
  - 53238895771820
  - 6189192838687
  - 48623327840257
  - 47264919314001
  - 42510070950746
  - 16878504505970
  - 22744978157662
  - 23644842894223
  - 71614018816334
  - 24651499733229
```

Листинг разработанной программы

main.rs

```
extern crate encoding_rs;
extern crate byteorder;
extern crate yaml_rust;
extern crate getopts;

mod rsa;

use getopts::Options;
use yaml_rust::YamlLoader;
use std::fs;
use std::env;

fn main() {
  // --- Adding cmd line arguments: ------------------
  //      -f: Path to input data yaml-file
  let args: Vec<String> = env::args().collect();

  let mut opts = Options::new();
  opts.optopt("f", "file", "input file path", "input.yaml");

  let matches = match opts.parse(&args[1..]) {
    Ok(m) => { m }
    Err(f) => { panic!(f.to_string()) }
  };

  if !matches.opt_present("f") {
    println!("You must specify the input file path!");
    return;
  }
  // -------------------------------------------------
  // --- Parsing input yaml-file ---------------------
  let file_contents = fs::read_to_string(matches.opt_str("f")
                                         .unwrap()).unwrap();
  let docs = YamlLoader::load_from_str(&file_contents).unwrap();
  let doc = &docs[0];
  let n = doc["N"].as_i64().unwrap();
  let e = doc["e"].as_i64().unwrap();
  // -------------------------------------------------
  // --- Finding d as a result of factorization of n -
  let d = rsa::get_d(n, e);
  // -------------------------------------------------
  // --- Decoding every present chunk of data --------
  let mut res = String::new();
  for c in doc["C"].as_vec().unwrap() {
    res.push_str(&rsa::decode_rsa(d, n, c.as_i64().unwrap()));
  }
  println!("\nDecoded text: {}", res);
```

```
48    // ------------------------------------------------
49  }
```

rsa.rs

```
1   #[path = "./mod_ops.rs"] mod mod_ops;
2   use mod_ops::{mod_exp, mod_inv};
3
4   // function that performs the factorization of n
5   pub fn get_d(n: i64, e: i64) -> i64 {
6     let mut i = 0;
7     let mut sqrt_w = (n as f64).sqrt(); // first candidate - sqrt(n)
8     let mut t = sqrt_w.round() as i64;
9     println!("n = {}", t);
10    println!("--------------------------------");
11    // keep moving until current w is not whole
12    while sqrt_w.fract() != 0.0 {
13      i += 1; // increment iteration counter
14      t += 1; // current t - increments every iteration
15      let w_i = (t * t) - n; // w_i = t^2 - n
16      println!("t_{} = {}; w_{} = {}", i, t, i, w_i);
17      sqrt_w = (w_i as f64).sqrt(); // count sqrt(w_i) to check is it whole
18    }
19    println!("--------------------------------");
20    let r_w = sqrt_w as i64; // resulting w converts to int
21    println!("t = {}; sqrt(w) = {}", t, r_w);
22    let p = t + r_w; // p = (t + sqrt(w))
23    let q = t - r_w; // q = (t - sqrt(w))
24    println!("p = {}; q =, {}", p, q);
25    let phi_n = (p - 1) * (q - 1); // Phi(n) = (p - 1)(q - 1)
26    println!("Phi(N) = {}", phi_n);
27    let d = mod_inv(e as isize, phi_n as isize) as i64; // d = e^-1 mod Phi(n)
28    println!("d = {}\n", d);
29    return d;
30  }
31
32  // function that decodes chunk of data based on given d and n
33  pub fn decode_rsa(d: i64, n: i64, c: i64) -> String {
34    // encoder for win1251 to support cyrillic symbols
35    let encoder = encoding_rs::WINDOWS_1251;
36    println!("C = {}", c);
37    let m = mod_exp(&c, &d, &n); // M = C^d mod N
38    println!("M = {}", m);
39    // convert M to symbols in win1251
40    let bs = m.to_bytes_be().1;
41    let (res, _, _) = encoder.decode(&bs);
42    println!("Message = {}", res);
43    return res.to_string();
44  }
```

Результаты работы программы

```
keker (~/code/itmo-4th-year/infosec/part2/lab1/target/debug) master ▲ 2
↳ ./lab1 -f ../../src/var10.yaml
n = 8776530
--------------------------------
t_1 = 8776531; w_1 = 19544412
t_2 = 8776532; w_2 = 37097475
t_3 = 8776533; w_3 = 54650540
t_4 = 8776534; w_4 = 72203607
t_5 = 8776535; w_5 = 89756676
--------------------------------
t = 8776535; sqrt(w) = 9474
p = 8786009; q =, 8767061
Phi(N) = 77027459296480
d = 8540915045653

C = 18937689886043
M = 4075692279
Message = то ч
C = 6667195679130
M = 3908168686
Message = исло
C = 53238895771820
M = 552592880
Message =  пер
C = 6189192838687
M = 3856982242
Message = едав
C = 48623327840257
M = 3773164795
Message = аемы
C = 47264919314001
M = 4112578792
Message = х ши
C = 42510070950746
M = 4042189550
Message = роко
C = 16878504505970
M = 3806722528
Message = веща
C = 22744978157662
M = 4075154428
Message = тель
C = 23644842894223
M = 3992712480
Message = ных
C = 71614018816334
M = 4024494821
Message = паке
C = 24651499733229
M = 4075741791
Message = тов_

Decoded text: то число передаваемых широковещательных пакетов_
```

Рис. 1: Результат работы программы

Вывод

В результате выполнения данной лабораторной работы была изучена атака на алгоритм шифрования RSA методом Ферма. Была реализована программа, позволяющая найти секретный ключ и расшифровать сообщение, зашифрованное с помощью RSA.