Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Кафедра информатики

Лабораторная работа №3

Ассиметричная криптография. RSA

Выполнил:
студент гр. 653503
Лисковец Б.Н.

Проверил:
В. С. Артемьев

Минск, 2019

Введение

RSA (аббревиатура от фамилий Rivest, Shamir и Adleman) — криптографический алгоритм с открытым ключом, основывающийся на вычислительной сложности задачи факторизации больших целых чисел.

Криптосистема RSA стала первой системой, пригодной и для шифрования, и для цифровой подписи. Алгоритм используется в большом числе криптографических приложений, включая PGP, S/MIME, TLS/SSL, IPSEC/IKE и других.

Задание: Реализовать программные средства шифрования и дешифрования текстовых файлов при помощи алгоритма RSA

Теоретическая часть

Криптографические системы с открытым ключом используют так называемые односторонние функции, которые обладают следующим свойством:

- если известно x, то $f(x)$ вычислить относительно просто;
- если известно $y = f(x)$, то для вычисления x нет простого (эффективного) пути.

Под односторонностью понимается не теоретическая однонаправленность, а практическая невозможность вычислить обратное значение, используя современные вычислительные средства, за обозримый интервал времени.

В основу криптографической системы с открытым ключом RSA положена сложность задачи факторизации произведения двух больших простых чисел. Для шифрования используется операция возведения в степень по модулю большого числа. Для дешифрования (обратной операции) за разумное время необходимо уметь вычислять функцию Эйлера от данного большого числа, для чего необходимо знать разложение числа на простые множители.

В криптографической системе с открытым ключом каждый участник располагает как открытым ключом (англ. public key), так и закрытым ключом (англ. private key). В криптографической системе RSA каждый ключ состоит из пары целых чисел. Каждый участник создаёт свой открытый и закрытый ключ самостоятельно. Закрытый ключ каждый из них держит в секрете, а открытые ключи можно сообщать кому угодно или даже публиковать их. Открытый и закрытый ключи каждого участника обмена сообщениями в криптосистеме RSA образуют «согласованную пару» в том смысле, что они являются взаимно обратными, то есть:

$\forall$ допустимых пар открытого и закрытого ключей (p, s)

$\exists$ соответствующие функции шифрования $E_p(x)$ и расшифрования $D_s(x)$ такие, что

$\forall$ сообщения $m \in M$, где M M — множество допустимых сообщений,

$$m = D_s(E_p(m)) = E_p(D_s(m))$$

Алгоритм создания открытого и секретного ключей

RSA-ключи генерируются следующим образом:

1. Выбираются два различных случайных простых числа p и q заданного размера (например, 1024 бита каждое).
2. Вычисляется их произведение $n = p \cdot q$, которое называется модулем.

3. Вычисляется значение функции Эйлера от числа n: $\varphi(n) = (p-1) \cdot (q-1)$.

4. Выбирается целое число $e1 < e < \varphi(n)$, взаимно простое со значением функции $\varphi(n)$.

   - Число e называется открытой экспонентой (англ. public exponent)
   - Обычно в качестве e берут простые числа, содержащие небольшое количество единичных бит в двоичной записи, например, простые из чисел Ферма: 17, 257 или 65537, так как в этом случае время, необходимое для шифрования с использованием быстрого возведения в степень будет меньше.
   - Слишком малые значения e, например 3, потенциально могут ослабить безопасность схемы RSA.

5. Вычисляется число d, мультипликативно обратное к числу e по модулю $\varphi(n)$, то есть число, удовлетворяющее сравнению: $d \cdot e \equiv 1 \pmod{\varphi(n)}$. Число d называется секретной экспонентой. Обычно оно вычисляется при помощи расширенного алгоритма Евклида.

6. Пара (e, n) публикуется в качестве открытого ключа RSA (англ. RSA public key).

7. Пара (d, n) играет роль закрытого ключа RSA (англ. RSA private key) и держится в секрете.

Шифрование и расшифрование

Предположим, Боб хочет послать Алисе сообщение m.
Сообщениями являются целые числа в интервале от 0 до $n-1$, т.е $m \in \mathbb{Z}_n$.
Алгоритм шифрования:

1. Взять открытый ключ (e, n) Алисы

2. Взять открытый текст m

3. Зашифровать сообщение с использованием открытого ключа Алисы: $c = E(m) = m^e \mod n$

Алгоритм расшифрования:

1. Принять зашифрованное сообщение c

2. Взять свой закрытый ключ (d, n)

3. Применить закрытый ключ для расшифрования сообщения: $m = D(c) = c^d \mod n$

Практическая часть

Для работы с программой задаётся два простых числа, необходимых для генерации ключей, а также задаётся сообщение для шифрования/расшифрования.

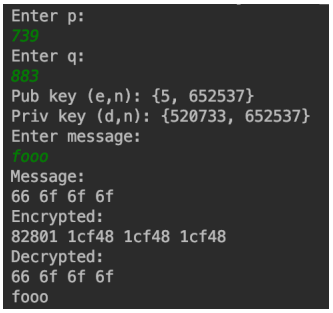

Рис. 1: Демонстрация работы программы

Код программы

```
u64 gcd(u64 a, u64 b) {
  if (a < b) {
    u64 t = a; a = b; b = t;
  }

  while (true) {
    u64 r = a % b;
    if (r == 0)
      return b;
    a = b;
    b = r;
  }
}

bool is_prime(u64 n) {
  u64 root = static_cast<u64>(floor(sqrt(static_cast<long double>(n))));
  for (u64 i = 2; i <= root; ++i) {
    if (n % i == 0)
      return false;
  }
  return true;
}

u64 calc_e(u64 t) {
  for (u64 e = 2; e < t; e++) {
    if (gcd(e, t) == 1) {
      return e;
    }
  }
  return -1;
}

u64 calc_d(u64 e, u64 t) {
  u64 k = 1;
  while (true) {
    k += t;
    if (k % e == 0) {
      return k / e;
    }
  }
}

std::vector<u64> encrypt(const std::vector<u64> &msg, u64 e, u64 n) {
  std::vector<u64> enc(msg.size(), 0);
  for (std::size_t k = 0; k < msg.size(); ++k) {
    u64 res = msg[k];
    for (u64 i = 0; i < e - 1; ++i) {
      res *= msg[k];
      res %= n;
```

```
    }
    enc[k] = res;
  }
  return enc;
}

std::vector<u64> decrypt(const std::vector<u64> &enc, u64 d, u64 n) {
  std::vector<u64> dec(enc.size(), 0);
  for (std::size_t k = 0; k < enc.size(); ++k) {
    u64 res = enc[k];
    for (u64 i = 0; i < d - 1; ++i) {
      res *= enc[k];
      res %= n;
    }
    dec[k] = res;
  }
  return dec;
}

int main() {
  // u64 p = 65537;
  // u64 q = 523;
  u64 p = prompt_prime("p"); // 65537
  u64 q = prompt_prime("q"); // 523

  u64 n = p * q;
  u64 t = (p - 1) * (q - 1);
  u64 e = calc_e(t);
  u64 d = calc_d(e, t);

  std::cout << "Pub key (e,n): {" << e << ", " << n << "}" << std::endl;
  std::cout << "Priv key (d,n): {" << d << ", " << n << "}" << std::endl;

  auto msg = prompt_msg();
  std::cout << "Message: " << std::endl;
  print_vector_uint(msg);

  auto enc = encrypt(msg, e, n);
  std::cout << "Encrypted: " << std::endl;
  print_vector_uint(enc);

  auto dec = decrypt(enc, d, n);
  std::string dec_msg(dec.begin(), dec.end());
  std::cout << "Decrypted: " << std::endl;
  print_vector_uint(dec);
  std::cout << dec_msg << std::endl;

  return 0;
}
```