

Министерство образования и науки Республики Беларусь
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ
Кафедра радиотехнических систем

МЕТОДИЧЕСКИЕ УКАЗАНИЯ
к лабораторной работе по дисциплине
ЗАЩИТА ИНФОРМАЦИИ
для студентов специальности
«Радиотехнические системы»

МИНСК 1999

УДК 681.322.067

Методические указания к лабораторной работе ИССЛЕДОВАНИЕ АЛГОРИТМА ЗАЩИТЫ ИНФОРМАЦИИ RSA по курсу «Защита информации» для студентов специальности «Радиотехнические системы»/ Сост. С.Б.Саломатин.- Мн.:БГУИР,1999- 22 с.

Методические указания содержат теоретические сведения, алгоритмы, порядок выполнения лабораторной работы, посвященной исследованию криптографическим системам защиты информации и открытым ключом. Исследуются криптографические алгоритмы RSA, ЭльГамала, Diffie-Hellman, алгоритмы электронной подписи, открытого распределения ключей, вероятностной атаки на алгоритм RSA. Лабораторная работа выполняется на ЭВМ с использованием программного пакета Maple.

Список лит - 4 назв.

Составитель С.Б.Саломатин

© Составление С.Б.Саломатин, 1998

1. ЦЕЛЬ РАБОТЫ

1. Изучить криптографические методы защиты информации с открытым ключом.
2. Исследовать алгоритмы распределения ключей.
3. Получить навыки программирования алгоритмов защиты информации с открытыми ключами.

2. КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

2.1. Криптосистемы с открытым ключом

Система с *открытым ключом* впервые была введена W.Diffie, M.E.Hellman в 1976 году. Суть данной криптографической системы заключается в следующем.

Каждый пользователь U имеет свой собственный алгоритм E_u шифрования и алгоритм дешифрования D_u (или получает его от достоверного источника). Алгоритмы шифрования и дешифрования сообщения m должны обладать следующими свойствами.

Свойство 1 . $D_u(E_u(m)) = m$ для любого сообщения m и для каждого пользователя U . Каждый пользователь U помещает алгоритм шифрования E_u в открытой книге. Алгоритм дешифрования D_u является секретным и хранится у пользователя U . Если пользователь A хочет послать сообщение m пользователю B , то он пользуется открытым алгоритмом E_B пользователя B и выполняет операцию

$$C = E_B(m), \quad (1)$$

после чего посылает зашифрованное сообщение C пользователю B .

Пользователь B дешифрирует принятое сообщение по правилу

$$D_B(C) = D_B(E_B(m)) = m. \quad (2)$$

Свойство 2. Алгоритмы E_U и D_U не требуют больших вычислительных затрат и памяти.

Свойство 3. Практически невозможно подобрать алгоритм D_u^* для E_u такой, чтобы $D_u^*(E_u(m)) = m$ для всех возможных m .

Авторы криптосистемы «открытый ключ» предложили для выполнения свойства 3 использовать так называемую **одностороннюю функцию**. Под односторонней понимается функция f , которую просто вычислить, но найти обратную к ней функцию f^{-1} крайне затруднительно.

Односторонняя функция может с успехом использоваться для *аутентификации* пользователя. Рассмотрим ситуацию, когда каждый пользователь имеет свой персональный идентификационный (PIN) код x_u . В устройстве идентификации (компьютере) хранится функция $y=f(x_u)$, которая отождествляется с именем пользователя U . Когда пользователь U хочет получить доступ, он должен передать свое имя и x_u . Компьютер вычисляет $f(x_u)$ и сравнивает $f(x_u)=y_u$. Если эти два значения совпадают, то пользователь получает доступ. В противном случае – нет. Преимуществом такой процедуры является то, что ключ не хранится в компьютере. Решение задачи аутентификации выдвинуло следующее требование.

Свойство 4. $E_u(D_u(m)) = m$ для всех пользователей u и для всех сообщений m .

Если пользователь A хочет послать сообщение m пользователю B , сопроводив это сообщение собственной сигнатурой, то он выполняет преобразование

$$C = D_A(m). \quad (3)$$

Пользователь B восстанавливает m из C с помощью открытого ключа E_A :

$$E_A(C) = E_A(D_A(m)) = m. \quad (4)$$

Свойство 5. Практически невозможно подобрать алгоритм D_A^* для E_u , который бы удовлетворял условию $E_u(D_u^*(m)) = m$ для всех возможных m .

Пользователь А с большой степенью вероятности может ожидать, что случайная выборка не приведет к ложной идентификации (4).

С другой стороны, любой пользователь может определить m из C , однако только пользователь А знает пару $(m, D_A(m))$.

Цифровая подпись. Алгоритм передачи пользователем А сообщения m и сигнатуры пользователю В имеет вид

- Открытый ключ – алгоритм E_u для всех пользователей U .
- Секретным является алгоритм D_u .
- Используются свойства 2,4,5.
- А передает В сообщение m путем преобразования $D_A(m)=C$.
- Пользователь В вычисляет $E_A(C)=m$.
- Сигнатурой является пара $(m, D_A(m))$.

Если пользователь А хочет передать В сообщение m и сигнатуру в зашифрованном виде, то он должен воспользоваться свойствами 1-5.

Процедура передачи в этом случае выглядит следующим образом:

- Открытым остается алгоритм E_u для всех пользователей.
- Секретным является алгоритм D_u .
- Используются свойства 1-5.
- Пользователь А передает В сообщение m после преобразования $E_B(D_A(m)) = C$.
- Пользователь В вычисляет $E_A(D_B(C)) = m$.
- Сигнатурой является пара $(m, D_B(C) = D_A(m))$.

Алгоритм цифровой подписи ЭльГамала. Для разрешения споров между отправителем и получателем информации, связанных с возможностью искажения ключа проверки подписи, достоверная копия этого ключа выдается третьей стороне «арбитру» и применяется им при

возникновении конфликта. Каждый может расшифровать сообщение, но так как ключ известен только отправителю, то никто другой, кроме него, не мог бы послать зашифрованное сообщение или подтвердить подпись..

Протокол подтверждения подлинности отправителя сообщения.

- Отправитель А и получатель В знают число p и случайное число N из интервала $(1, N)$. А генерирует случайные числа x и y из того же интервала.

- x является секретным ключом, а y должно быть взаимно простым с $(p-1)$.

- Пользователь А вычисляет $Q = N^x \bmod p$ и $R = N^y \bmod p$ и решает относительно S уравнение

$$T = xR + yS \bmod (p-1),$$

передает В документ с подписью (Q, R, S, T) .

- Получатель В проверяет подпись, контролируя тождество $A^S = B^R \cdot R^T \bmod p$.

В этой системе секретным ключом для подписания сообщений является число x , а открытым ключом для проверки достоверности подписи - число Q .

2.2. Криптосистема RSA

В 1978 году R.L.Rivest, A.Shamir, L.Adleman опубликовали криптосистему с открытым ключом, которая стала известной как RSA система. Рассматриваемая система криптографии основывается на следующей теореме.

Теорема Эйлера. Пусть a и n целые числа. Тогда

$$\text{НОД}(a, n) = 1 \Rightarrow a^{\phi(n)} \equiv 1 \bmod n, \quad (5)$$

где НОД (а, n) – наибольший общий делитель чисел а и n, $\phi(n)$ - функция Эйлера

$$\phi(n) = |\{1 \leq i \leq n \mid \text{НОД}(i, n) = 1\}| = n \cdot \prod_{p|n} (1 - \frac{1}{p}) \quad (6)$$

определяется для всех целых положительных n и представляет собой число чисел ряда $0, 1, \dots, n - 1$ взаимно простых с n.

Если существует каноническое разложение числа n

$$n = p_1^{\alpha_1} p_2^{\alpha_2} \dots p_k^{\alpha_k},$$

тогда

$$\phi(n) = n(1 - \frac{1}{p_1})(1 - \frac{1}{p_2}) \dots (1 - \frac{1}{p_k}) = (p_1^{\alpha_1} - p_1^{\alpha_1-1})(p_2^{\alpha_2} - p_2^{\alpha_2-1}) \dots (p_k^{\alpha_k} - p_k^{\alpha_k-1}).$$

Алгоритм RSA. Каждый пользователь выбирает два простых числа p_u и q_u , произведение которых образует число $n_u = p_u \cdot q_u$. Из (2) следует, что

$$\phi(n_u) = (p_u - 1) \cdot (q_u - 1). \quad (7)$$

Далее пользователь U определяет целое число e, $1 < e_u < \phi(n_u)$, для которого $\text{НОД}(e_u, \phi(n_u)) = 1$. С помощью алгоритма Евклида пользователь U вычисляет за $2 \log_2 \phi(n_u)$ шагов целое число d_u , удовлетворяющее сравнению

$$e_u \cdot d_u \equiv 1 \pmod{\phi(n_u)}, \quad 1 < d_u < \phi(n_u). \quad (8)$$

Пользователь U публикует числа e_u и n_u в открытой книге, но держит d_u в секрете. Числа p_u и q_u не публикуются.

Шифрование сообщения. Если пользователь А хочет передать сообщение m пользователю В ($0 < m < n_B$), то А выбирает из открытой книги число e_B и вычисляет зашифрованный текст:

$$C \equiv m^{e_B} \pmod{n_B}. \quad (9)$$

Расшифровка сообщения. Пользователь В восстанавливает сообщение m из C, вычисляя

$$P \Rightarrow C^{d_B} \equiv m^{e_B \cdot d_B} \equiv m^{1+\phi(n_B)} \equiv m \pmod{n_B}; \quad (l - \text{любое}), (10)$$

где НОД(m, n_B)=1.

Пример 1. Пусть

$$p=211, q=223, n = p \cdot q = 47053, \phi(n) = (p-1)(q-1)=46620,$$

открытый ключ $e = 16813$, секретный ключ – $d = 19837$.

Пользователь хочет передать английский текст RSA. Буквам соответствуют следующие номера их положения в английском алфавите: R=18, S=19, A=1. Тогда 5- битовое представление текста будет иметь следующий вид:

$$m = ((1 \cdot 32) + 19) \cdot 32 + 18 = 1650.$$

После шифрования получаем

$$C \equiv m^d \pmod{n} = 1650^{16813} \pmod{47053} = 3071.$$

Получатель расшифровывает сообщение с помощью секретного ключа:

$$P \equiv C^e \pmod{n} = 3071^{19837} \pmod{47053} = 1650.$$

Пример 2. Рассмотрим случай, когда используется N - значный алфавит и известны положительные числа k и l , $k < l$, причем значения N^k и N^l охватывают диапазон около 200 значного числа. Используется английский алфавит из $N = 26$ букв, $k = 3$, $l = 4$. Шифруемое слово “YES”.

Формирование ключей пользователя A

1. Определяется величина модуля $\cdot N^k < n_A < N^l$. Для этого выбираются простые числа $p_A = 281$, $q_A = 167$, $n_A = p_A \cdot q_A = 281 \cdot 167 = 46927$.
2. Используя генератор случайных чисел определяется открытый ключ $e_A = 39423$, удовлетворяющий условиям $\text{НОД}(e_A, p_A-1) = \text{НОД}(e_A, q_A-1) = 1$.
3. Вычисляется секретный ключ $d_A = e_A^{-1} \pmod{(p_A-1)(q_A-1)} = 26767$.
4. Числа p_A , q_A , d_A являются секретными.

Пользователь В использует для шифрования ключ $(n_A, e_A) = (46927, 39423)$. Сообщение представляется в цифровом виде как триграф следующим образом:

$$24 \cdot 26^2 + 4 \cdot 26 + 18 = 16346.$$

После шифрования алгоритмом RSA получаем $16346^{39423} \bmod 46927$, что дает число 21166.

Пользователю А известен свой секретный ключ $(n_A, d_A) = (46927, 26767)$, и для расшифровки полученного сообщения А использует преобразование

$$21166^{26767} \bmod 46927 = 16346 = \text{“YES”}.$$

Наиболее сложным шагом с вычислительной точки зрения в проделанных выше действиях является операция вычисления модулярной экспоненты $16346^{39423} \bmod 46927$. Если использовать метод последовательного возведения числа в квадрат, то алгоритмическая сложность может быть оценена величиной $O(t^3)$, где t – количество бит в числе.

Система RSA

Открытый ключ	e_u n_u для всех пользователей U
Секретный ключ	d_u
Свойства	$e_u d_u \equiv 1 \bmod \phi(n_u)$
Сообщение от А к В	$0 < m < n_B$
Шифрование сообщения пользователем А	$C \equiv m^{e_A} \bmod n_B$
Расшифровка сообщения пользователем В	$P \Rightarrow C^{d_B} \equiv m^{e_A \cdot d_B} \equiv$ $\equiv m^{1 + l\phi(n_B)} \equiv m \bmod n_B; \quad (l - \text{любое})$

Шифр ЭльГамала. Схема шифрования основана на возведении в степень по модулю большого простого числа p . Сообщения представляются целыми числами m из интервала $(1, p)$.

Протокол сообщения m выглядит следующим образом:

- Отправитель A и получатель B знают лишь число p . A генерирует случайное число $x \in (1, p)$, B тоже генерирует случайное число y из того же интервала.
- A шифрует сообщение $C_1 = m^x \bmod p$ и посылает его B .
- B шифрует его своим ключом $C_2 = C_1^y \bmod p$ и посылает C_2 к A .
- A снимает свой ключ, выполняя преобразование $C_3 = C_2^{x^{-1}} \bmod p$, и возвращает C_3 пользователю B .
- Получатель B расшифровывает сообщение $C_3^{y^{-1}} = m \bmod p$.
- Обратные числа x^{-1} и y^{-1} могут быть найдены с помощью алгоритма Евклида.

2.3. Открытое распределение ключей

На основе методов шифрования с открытым ключом легко решать задачу выработки общего секретного ключа для сеанса связи любой пары пользователей.

Алгоритм W. Diffie, M. E. Hellman. Подразумевает независимое генерирование каждым из пары связывающихся пользователей своего случайного числа, преобразование его посредством некоторой процедуры, обмен преобразованными числами по открытому каналу связи и вычисление общего секретного ключа на основе информации, полученной в процессе связи от партнера.

Процедура открытого распределения ключей между абонентами A и B выглядит следующим образом:

- Абоненты А, В знают два открытых числа p и α . Число p является большим простым числом, $(p-1)$ – имеет, по крайней мере, один большой множитель. В качестве числа α может использоваться примитивный элемент поля $GF(p)$.
- Пользователь А имеет свой секретный ключ d_A , а пользователь В – секретный ключ d_B , $d_A, d_B \in (1, N)$.
- Абонент А посылает В шифровку своего ключа $Z^\bullet = \alpha^{d_A} \bmod p$, а абонент В посылает А шифровку своего ключа $Z^{\bullet\bullet} = \alpha^{d_B} \bmod p$.
- Общий секретный ключ вычисляется, как

$$K = Z^{\bullet d_B} = Z^{\bullet\bullet d_A} = \alpha^{d_A d_B} \bmod p.$$

2.4. Алгоритм вероятностной атаки на криптосистему RSA

Из алгоритма RSA следует, что число n определяется как произведение двух чисел $(p \cdot q)$ и знание функции Эйлера $\phi(n)$ эквивалентно знанию сомножителей факторизации n . Предположим, что ставится задача «взломать» систему RSA, т.е. определить положительное число d , удовлетворяющее условию

$$a^{ed} \equiv a \bmod n$$

для всех a простых к n . Это эквивалентно тому, что $(ed - 1)$ становится кратным НОК чисел $(p-1)$, $(q-1)$.

Предположим, что известно n (являющееся произведением двух простых чисел) и также некое целое число m , удовлетворяющее сравнению $a^m \equiv 1 \bmod n$ для всех a взаимно простых с n . Заметим, что в этом случае m должно быть четным числом, поэтому, не меняя существа задачи, можно вместо m в дальнейшем использовать число $m/2$. Отсюда следует, что выражение $a^{m/2} \not\equiv 1 \bmod n$ справедливо, по крайней мере, для 50% a из

$(\mathbb{Z}/n\mathbb{Z})$. Таким образом, если осуществляется тестирование множества случайно выбранных $\{a_i\}$ и определяется, что во всех случаях выполняется сравнение $a^{m/2} \equiv 1 \pmod{n}$, тогда с большой степенью вероятности можно утверждать, что все a и n взаимно просты. При этом возможны следующие варианты:

- $m/2$ кратно одному из двух чисел $(p-1)$ или $(q-1)$ (для определенности положим p), но не кратно обоим этим числам. В этом случае всегда выполняется сравнение $a^{m/2} \equiv 1 \pmod{p}$, но в 50% попытки сравнения по модулю q будут приводить к (-1) .
- $m/2$ не кратно любому из двух чисел $(p-1)$ или $(q-1)$. В этом случае сравнения $a^{m/2} \equiv 1 \pmod{p}$ или q (и следовательно, n) выполняются в 25% попыток; сравнение $a^{m/2} \equiv -1 \pmod{p}$ или q выполняется в 25% попыток и в оставшиеся 50% выполняются сравнения a по модулю других простых чисел.

2.5. Алгоритм Евклида

Наибольшим общим делителем (НОД) называется всякое целое, делящее одновременно целые числа a, b, \dots, l . Обозначается наибольший общий делитель символом $\text{НОД}(a, b, \dots, l)$. Для нахождения НОД чисел, а также для нахождения обратного числа часто используется алгоритм Евклида. Его суть состоит в следующем.

Пусть a и b два положительных целых числа $b \geq a$. Делитель a и b является также делителем a и $(b-a)$, и наоборот, $\text{НОД}(a, b) = \text{НОД}(a, b-a)$.

В свою очередь число b можно представить в следующем виде:

$b = q \cdot a + r, \quad 0 \leq r \leq a$, таким образом, $\text{НОД}(a, b) = \text{НОД}(r, a)$, если $r=0$, то $\text{НОД}(a, b) = a$.

Алгоритм вычисления НОД

Пусть a и b – положительные целые. Тогда можно составить следующий ряд равенств:

$$a = bq_1 + r_2, \quad 0 < r_2 < b,$$

$$b = r_2q_2 + r_3, \quad 0 < r_3 < r_2,$$

$$r_2 = r_3q_3 + r_4, \quad 0 < r_4 < r_3,$$

$$\dots\dots\dots$$

$$r_{n-2} = r_{n-1}q_{n-1} + r_n, \quad 0 < r_n < r_{n-1},$$

$$r_{n-1} = r_nq_n,$$

заканчивающийся, когда получаем некоторое $r_{n+1}=0$. Рассматривая приведенные выше равенства, можно убедиться, что общие делители чисел a и b одинаковы с общими делителями чисел b и r_2 , далее одинаковы с общими делителями чисел r_2, r_3 , чисел r_3, r_4, \dots , чисел r_{n-1} и r_n , наконец, с делителем одного числа r_n .

Отсюда следует, что совокупность общих делителей чисел a и b совпадает с совокупностью делителей их НОД. Этот НОД равен r_n , т.е. последнему не равному нулю остатку алгоритма Евклида.

Для нахождения обратного числа алгоритм Евклида следует прочитать наоборот.

Пример. Используя алгоритм Евклида найти обратное число 437^{-1} по модулю 729. Предварительно применим алгоритм Евклида к числам 729 и 437

$$729 = 437 + 292;$$

$$437 = 292 + 145;$$

$$292 = 2 \cdot 145 + 2;$$

$$145 = 72 \cdot 2 + 1.$$

$$\text{НОД}(729, 437) = 1.$$

Для нахождения обратного числа применим следующую процедуру:

$$1 = 145 - 72 \cdot 2;$$

$$\begin{aligned}
&=145 - 72(292 - 2 \cdot 145); \\
&=145 \cdot 145 - 72 \cdot 292; \\
&=145(437-292)-72 \cdot 292; \\
&=145 \cdot 437 - 217 \cdot 292; \\
&=145 \cdot 437 - 217(729-437); \\
&\equiv 362 \cdot 437 \bmod 729.
\end{aligned}$$

Обратное число $437^{-1} = 362$.

2.6. Алгоритм вычисления модулярной экспоненты

Метод последовательного возведения числа в квадрат

Пусть число k имеет двоичное представление следующего вида:

$$\sum_{i=0}^t k_i \cdot 2^i = k_0 + k_1 \cdot 2 + k_2 \cdot 2^2 + \dots + k_t \cdot 2^t, \quad k_i \in \{0,1\}$$

Тогда

$$a^k = \prod_{i=0}^t a^{k_i 2^i} = (a^{2^0})^{k_0} (a^{2^1})^{k_1} \dots (a^{2^t})^{k_t}.$$

Алгоритм

Вход: $a \in \mathbb{Z}_n$, и целое число $0 \leq k < n$, имеющее двоичную форму

представления $k = \sum_{i=0}^t k_i 2^i$.

Выход: $a^k \bmod n$.

1. Положить $b \leftarrow 1$. Если $k=0$, тогда вернуться к (b).
2. Положить $A \leftarrow a$.
3. Если $k_0 = 1$ тогда положить $b \leftarrow a$.
4. Для i от 1 до t делать следующие операции:
5. Положить $A \leftarrow A^2 \bmod n$.
6. Если $k_i = 1$, тогда положить $b \leftarrow A \cdot b \bmod n$
7. Вернуться к (b).

Пример. Вычислить модулярную экспоненту $5^{596} \bmod 1234 = 1013$.

Процесс вычисления иллюстрируется таблицей

i	0	1	2	3	4	5	6	7	8	9
k	0	0	1	0	1	0	1	0	0	1
i	A	5	2	6	6	1	3	4	7	9
		5	25	81	011	69	21	79	47	25
	b	1	1	6	6	6	6	1	1	1
			25	25	7	7	059	059	059	013

Сложность вычисления модулярной экспоненты по данному алгоритму оценивается выражением $O((\lg n)^3)$.

3. ОПИСАНИЕ ПРОГРАММНОГО ПАКЕТА MAPLE

Программный пакет Maple состоит из ядра – процедур, написанных на языке C, и библиотеки, написанной на Maple- языке и интерфейса. Ядро выполняет большинство базисных операций. Библиотека содержит множество команд – процедур, выполняемых в режиме интерпретации. Программируя собственные процедуры, пользователь может пополнять ими стандартный набор возможностей Maple. Основные команды приведены в справочнике по Maple, который выдается студенту при выполнении данной лабораторной работы.

Пакет Maple имеет много команд для эффективной работы в теории чисел. Некоторые из этих команд находятся в стандартной библиотеке, но большинство содержатся в трех библиотеках: numtheory, GaussIn и padic.

Входными параметрами и результатами действий этих команд выступают числа – целые, рациональные, цепные дроби, гауссовские целые (комплексные целые), p-адические, простые, а также несколько известных иррациональных чисел.

Используя команды Maple, достаточно просто моделировать криптографические алгоритмы защиты информации. В качестве примера

ниже даются программы для моделирования основных алгоритмов, исследуемых в лабораторной работе.

Алгоритм шифрования RSA (версия Maple 5)

Функция to number

```
>to_number:=proc(st)
>local ll,nn,ss,ii,num;
>num:=table(['a'=1,'b'=2,'c'=3,'d'=4,'e'=5,'f'=6,'g'=7,'h'=8,'i'=9,'j'=10,'k'=11,'l'=12,
'm'=13,'n'=14,'o'=15,'p'=16,'q'=17,'r'=18,'s'=19,'t'=20,'u'=21,'v'=22,'w'=23,'x'=24,'
y'=25,'z'=26,' '=27]);
>if not type(st,string) then ERROR('wrong number (or type) of arguments') fi;
>ll:=length(st);
>if ll= 0 then RETURN(0) fi;
>nn:=1;
>for ii from 1 to ll do
>ss:=num[substring(st,ii..ii)];
>if(not type(ss,numeric)) then ERROR('wrong number (or type) of arguments')
fi;
>nn:=100*nn+ss;
>od;
>nn-10^(2*ll);
>end;
```

Функция from number

```
>from_number:=proc(nn)
>local ss,mm,ll,ii,ans,a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,y,z,' ',alpha;
>alpha:=table([1=a,2=b,3=c,4=d,5=e,6=f,7=g,8=h,9=i,10=j,11=k,12=l,13=m,14
=n,15=o,16=p,17=q,18=r,19=s,20=t,21=u,22=v,23=w,24=x,25=y,26=z,27=' ']);
>mm:=nn;
>if(not type(nn,integer)) then ERROR('wrong number (or type) of arguments')
fi;
>ll:=floor(trunc(evalf(log10(mm)))/2)+1;
>ans:='';
>for ii from 1 to ll do
>mm:=mm/100;
>ss:=alpha[frac(mm)*100];
>if(not type(ss,string)) then ERROR('wrong number (or type) of arguments') fi;
>ans:=cat(ss, ans);
>mm:=trunc(mm)
>od;
>ans;
```


>end:

Действие функции to number

>to_number('maple');

1301161205

Действие функции from number

>from_number(805121215);

hello

Определение больших простых чисел p и q

>p:=nextprime(7347124781320478301247);

p:=7347124781320478301377

>q:=nextprime(478574839027542389055784235534782043);

q:=478574839027542389055784235534782067

Определение числа n

>n:=p*q;

n:=3516149059535715479653013539650200318590903767034041006259

Определение функции Эйлера phi_n

>phi_n:=(p-1)*(q-1);

phi_n:=3516149059535715479652534964811172768854723201478027922816

Случайный выбор числа e из интервала от 0 до phi_n-1

>e:=nextprime(432432);

e:=432433

>evalf(phi_n/e);

.8131084028 10⁵²

Определение d из уравнения ed=1 mod phi_n

>igcdex(e,phi_n,'d','k');

>d:=d mod phi_n;

d:=2185903712533883860716989527152372758252773961337230693713

Ввод сообщения

>M:=to_number('hi');

M:=809

Шифрование сообщения

>C:=Power(M,e) mod n;

C:=445306247884178784227069275130014327352175968049363742217

Расшифровка сообщения

>Power(C,d) mod n;

809

>from_number("");

hi

>M:=to_number('i think therefore i am');

M:=9272008091411272008051805061518052709270113

>C:=Power(M,e) mod n;

C:=1130768087962955647611115745992731338735951488133460302383

>Power(C,d) mod n;

9272008091411272008051805061518052709270113

>from_number("");

i think therefore i am

Алгоритм шифрования Diffie-Hellman

Описание функции to_number

```
> to_number:=proc(st)
> local ll,nn,ss,ii,num;
> num:=table(['a'=1,'b'=2,'c'=3,'d'=4,'e'=5,'f'=6,'g'=7,'h'=8,'i'=9,'j'=10,'k'=
11,'l'=12,'m'=13,'n'=14,'o'=15,'p'=17,'r'=18,'s'=19,'t'=20,'u'=22,'w'=23,'x'=
24,'y'=25,'z'=26,'`=27']):
> if not type (st, string) then ERROR ('Wrong number (or type) of arguments')
fi;
> ll:= length (st);
> if ll=0 then RETURN(0) fi;
> nn:=1;
> for ii from 1 to ll do
> ss:= num[substring(st,ii..ii)];
> if (not type (ss,numeric)) then ERROR ('Wrong number (or type) of
arguments')fi
> nn:=100*nn+ss;
> od;
> nn-10^(2*ll);
> end;
```

Описание функции from_number

```
> from_number:=proc(nn)
> local ss,mm,ll,ii,ans,a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,y,z,`,alpha;
> alpha:=table([1=a,2=b,3=c,4=d,5=e,6=f,7=g,8=h,9=i,10=g,11=k,12=l,13=m,14
=n,15=p,17=q,18=r,19=s,20=t,21=u,22=v,23=w,24=x,25=y,26=z,27=`]):
> mm:=nn
> if (not type (nn,integer)) then ERROR('Wrong number (or type) of
arguments')fi;
> ans:=``;
> for ii from 1 to ll do
> mm:=mm/100;
> ss:=alpha[frac(mm)*100];
> if (not type(st,string)) then ERROR ('Wrong number (or type) of
arguments')fi;
> ans:=cat(ss,ans);
> od;
> ans;
```

>end:

Ключ пользователя А

>xA:=1234567;

$x_A := 12345$

Ключ пользователя В

>xB:=2345678;

$x_B := 2345678$

Большое простое число, причем (p-1) имеет, по крайней мере, один простой большой множитель

>p:=nextprime (343434536478990035836);

$p := 343434536478990035837$

Число (p-1) должно иметь , по крайней мере, один простой большой множитель

>ifactor (p-1);

$(2)^2(149)(576232443756694691)$

Секретный ключ, передаваемый пользователю В

>yA:=Power(a,xA) mod p;

$y_A := 247905479563139068073$

Секретный ключ, передаваемый пользователю А

>yB:=Power (a,xB) mod p;

$y_B := 38487642015460917674$

Выбор случайного числа из интервала 0 – (p-1)

>k:= 4435257;

$k := 4435257$

Вычисление ключа

>K:=Power(yB,k) mod p;

$K := \text{modp}(\text{Power}(y_B, 163600),)$

Ввод сообщения

>m:=to_number(`anatoly`);

$m := 1140120151225$

Шифрование сообщения парой (C1,C2)

>C1:=Powr(a,k) mod p;

$C1 := 11153774205254515333$

>C2:=K*m mod p;

$C2 := 183646741198931219975$

Расшифровка сообщения

>KD:=Power(C1,xB) mod p;

$KD:=294881765299568050502$
>mD:=C2/KD mod p;
 $mD:=1140120151225$
>from_number (mD);

anatoly

Версии программ для Maple 7 помещены в Приложении.

4. СОДЕРЖАНИЕ И ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

4.1. Содержание работы

1. Изучить принципы криптографических систем защиты информации с открытым ключом.
2. Изучить принципы и правила использования программного пакета Maple.
3. Изучить принципы, алгоритмы и провести моделирование на ЭВМ криптографической системы RSA
4. Изучить алгоритмы и провести моделирование на ЭВМ цифровой подписи (вычисление сигнатуры).
5. Изучить принципы организации, алгоритмы и провести моделирование на ЭВМ системы открытого распределения ключей.
6. Изучить метод и провести моделирование вероятностной атаки на криптографическую систему RSA.

4.2. Порядок выполнения работы

4.2.1. Исследование системы RSA

1. Получить у преподавателя вариант схемы шифрования системы RSA, данные p , q , текст сообщения и схему его цифрового преобразования.
2. Используя генератор случайных чисел, определить секретный ключ d .
3. Вычислить открытый ключ e .

4. Провести моделирование алгоритма RSA, зашифровав и расшифровав заданное сообщение.
5. Составить схему электронной подписи.
6. Провести моделирование алгоритма электронной подписи.

4.2.2. Исследование системы ЭльГамала

1. Получить у преподавателя вариант схемы шифрования системы ЭльГамала, исходные данные – число p , текст сообщения m .
2. Используя генератор случайных чисел, определить числа x , y .
3. Используя алгоритм Евклида, определить обратные числа x^{-1}, y^{-1} .
4. Провести моделирования криптосистемы ЭльГамала, зашифровав и расшифровав сообщение m .

4.2.3. Исследование системы открытого распределения ключей

1. Определить два открытых числа p , a .
2. Определить секретные ключи пользователей.
3. Вычислить общий секретный ключ
4. Используя общий секретный ключ как гамма шифр, зашифровать и расшифровать сообщение m .

4.2.4. Исследование алгоритма вероятностной атаки на криптосистему RSA.

1. Получить у преподавателя вариант схемы криптосистемы RSA.
2. Используя алгоритм, изложенный в п.2.4, осуществить вероятностную атаку для нахождения секретного ключа.
3. Оценить сложность проведенной атаки.

5. СОДЕРЖАНИЕ ОТЧЕТА

- Формулировка цели работы.
- Схемы исследуемых криптосистем.
- Результаты теоретических расчетов.
- Результаты моделирования на ЭВМ.

- Выводы и замечания по работе

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Поясните принцип криптосистемы RSA.
2. Каким образом определить обратное число?
3. Поясните принцип алгоритма ЭльГамала.
4. Составьте схему открытого распределения ключей.
5. Дайте определение обратной функции.
6. Каким образом алгоритм RSA может быть использован для создания электронной подписи?

ЛИТЕРАТУРА

1. Виноградов И.М. Основы теории чисел.- М.; Наука, 1972.
2. Henk C.A van Tilborg An Introduction to Cryptology.- Rluwer Academic Publishers, 1988.
3. Жельников В. Криптография от папируса до компьютера.-М., АБФ, 1996.
4. Говорухин В.Н., Цибулин В.Г. Введение в Maple. Математический пакет для всех.- М.: Мир, 1997.

ИССЛЕДОВАНИЕ СИСТЕМА RSA

(версия для Maple 7)

> restart;

1. Формирование больших простых чисел (разрядность 80 знаков)

> M1 := rand(10^80)();

M2 := rand(10^80)();

P1 := nextprime(M1);

P2 := nextprime(M2);

M1 := 196690813211106932703436330736974742561435635584587189767467538 \

30538032062222085

M2 := 741217686043056139217455800374092598119526553100754871637971794 \

90457039169594160

P1 := 196690813211106932703436330736974742561435635584587189767467538 \

30538032062222257

P2 := 741217686043056139217455800374092598119526553100754871637971794 \

90457039169594213

2. Фиксирование выбранных чисел

> M1 :=

19669081321110693270343633073697474256143563558458

718976746753830538032062222085;

> M2 :=

74121768604305613921745580037409259811952655310075

487163797179490457039169594160;

> P1 :=

19669081321110693270343633073697474256143563558458

718976746753830538032062222257;

> P2 :=

74121768604305613921745580037409259811952655310075

487163797179490457039169594213;

3. Вычисление модуля n и числа Эйлера n2

> n := P1 * P2;

> n2 := (P1 - 1) * (P2 - 1);

n := 14579070943426365719341081596858629803265159149118248616433975229 \

80497550736230615496046802186876835611836753440525199587698019954 \

839165932427842278373706998741

n2 := 1457907094342636571934108159685862980326515914911824861643397522 \

98049755073623052170519687677056964352262364233379113149147915142 \

0633025388494521283302475182272

4. Формирование открытого ключа e

4.1. *I – вариант.* Выбор ключа e как числа Ферма и проверка его на простоту (оператор isprime, gcd)

```
> e := 2^16+1; isprime(e); gcd(e, n2);  
e := 65537  
  
true  
  
1
```

4.2. *II – вариант.* Вычисление открытого ключа e с помощью генератора случайных чисел

```
e := rand(10^80)();
```

5. Вычисление ключа d как обратного числа к числу e по модулю $n2$

```
> d := eval(1/e mod n2);  
d := 34180298922096847472065507840720943425419102236324807359431775852 \  
71731215506077782931832401785220954991090874537848960948254750992 \  
26794560236481979918863102913
```

6. Алгоритмы шифрования encoder и расшифрования decoder

```
> encoder := (a, e, n) -> Power(a, e) mod n;  
  
> decoder := (a, d, n) -> Power(a, d) mod n;
```

Пример шифрования и расшифрования числа $a=6$

```
> m1 := encoder(6, e, n); p1 := decoder(m1, d, n);  
m1 := 458888351129631987399420997962608442637750006103185185055867041 \  
57170156910633316653024959038494946006288636768291532268510061345 \  
2421921948098948354938654783173  
  
p1 := 6
```

7. Шифрование и расшифрование сообщения

7.1. Создание сообщения message.

```
> mess1 := `Applications Digital signatures and  
envelopes`;  
mess1 := Applications Digital signatures and envelopes
```

7. 2. Алгоритм оцифровки сообщения. в коде стандарта ASCII. Использование формата ASCII позволяет просто отобразить текстовое сообщение в цифровой код и наоборот

```
> read "shortconverter.m";
```

Пример оцифровки

```
> messnum1 := shortconverter(mess1);
```



```
messnum1 := 600339000229473873243952841366938540166098639631014180176 \
108441864766195056138854696377780929317962683278707
```

Пример шифрования и расшифрования

```
> m2 := encoder(messnum1, e, n); p1 := decoder(m2, d, n);
m2 := 108413720440667038977328189499124249845870525584953082067704863 \
90474440013572006320676970614901504753491205499827596810533570677 \
66033877599058580207792460227378

p1 := 6003390002294738732439528413669385401660986396310141801761084418 \
64766195056138854696377780929317962683278707
```

8. Оценка длины сообщения (оператор length)

```
> length(m2);

160

4170706C69636174696F6E730A4469676974616C207369676E61747572657320616 \
E6420656E76656C6F706573
```

9. Шифрование и расшифрование с использованием специальных подпрограмм `getPublicKey.m`, `RSAL.m`, `encoderRSA.m`, `decoderRSA.m`. Подпрограммы инициализируются с помощью оператора «read».

9.1. Процедура формирования параметров (nR , ключей eR и dR) - криптосистемы RSA по заданной длине ключа L

```
> read "getPublicKey.m";
> read "RSAL.m";
> L := RSAL(200);
> nR := op(1, L);
> eR := op(2, L);
> dR := op(3, L);
L := [4088916452743014757436137997396015339749963198436415902073695346 \
84360421879105946044799744970243499910548981258967774757898713259 \
88885799190773101322351516314772919542262889007266858242480320872 \
196437, 36172445676785584019823118736745534092066888914143571109441 \
83633775803123657779387252452149858938711051798540795469857714665 \
54066201010172235890380314684175894788717205024120552379153330238 \
59075928257, 270584034482105648874603527696729848578662077008247910 \
06205372493999920350847578033639432348807936180396610096577850788 \
26510645531207565281017818663896088393313086239629590394516140928 \
5823972483256673]

nR := 408891645274301475743613799739601533974996319843641590207369534 \
68436042187910594604479974497024349991054898125896777475789871325 \
9888579919077310132235151631477291954226288900726685824248032087 \
2196437
```

```
eR := 361724456767855840198231187367455340920668889141435711094418363 \
37758031236577793872524521498589387110517985407954698577146655406 \
62010101722358903803146841758947887172050241205523791533302385907 \
5928257
```

```
dR := 270584034482105648874603527696729848578662077008247910062053724 \
93999920350847578033639432348807936180396610096577850788265106455 \
31207565281017818663896088393313086239629590394516140928582397248 \
3256673
```

9.2. Процедуры шифрования и расшифрования с использованием файлов библиотеки

```
> read "encoderRSA.m"; read "decoderRSA.m";
```

Пример

```
> g:=encoderRSA(messnum1, eR, nR) ;
g := 94541047817999973744048333725292340106076712531043741519907097492 \
63446965195000683814874869110223511139675660850703419517530671437 \
59203994647294139329589850119774137538740868232658518942113901370 \
8459
```

```
> decoderRSA(g, dR, nR) ;
60033900022947387324395284136693854016609863963101418017610844186476 \
6195056138854696377780929317962683278707
```

10. Процедура преобразования цифрового сообщения из формата ASCII в текст с помощью подпрограммы `numtstring.m`.

```
> read "numtstring.m";
```

Пример преобразования

```
> numtstring(p1) ;
"Applications Digital signatures and envelopes"
```

Задание

1. Определите систему RSA, задав открытый и закрытый ключи, модулю шифрования.
2. Ответьте на вопрос: «Сообщение какой максимальной длины Вы сможете обработать полученной системой RSA?»
3. Введите сообщения из M символов и закодируйте системой RSA
4. Докажите, что вы можете расшифровать сообщение с помощью закрытого ключа
5. Обменяйтесь открытыми ключами Вашей система RSA с Вашими коллегами по классу, зашифруйте и пошлите им сообщение, а также примите и расшифруйте их послание.

Св. план 1999, поз.63

Учебное издание

МЕТОДИЧЕСКИЕ УКАЗАНИЯ
к лабораторной работе
ИССЛЕДОВАНИЕ АЛГОРИТМА ЗАЩИТЫ ИНФОРМАЦИИ RSA
по курсу
ЗАЩИТА ИНФОРМАЦИИ
для студентов специальности
«Радиотехнические системы»

Составитель Саломатин Сергей Борисович

Редактор Н.В.Гриневич

Подписано в печать		Формат 60×84/16
Бумага	Печать офсетная	Усл.печ.л
Уч-изд. л. 1,0	Тираж 100 экз.	Заказ

Белорусский государственный университет информатики и
радиоэлектроники

Отпечатано в БГУИР. Лицензия ЛП №156. 220027, Минск, П. Бровки, 6