Лабораторная работа № 3. Криптографические системы с открытым ключом (4 часа)

3.1 ЦЕЛЬ РАБОТЫ

Изучение криптографических алгоритмов с открытым ключем.

3.2 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

Первые криптографические системы с открытым ключом или ассиметричные криптосистемы появились в конце 1970-х годов. От классических симметричных алгоритмов они отличаются тем, что для шифрования данных используется один ключ, обычно его называют открытый или публичный ключ, а для дешифрования – другой, секретный или закрытый, ключ.

Диффи и Хеллман, которые впервые предложили и описали криптосистему с открытым ключом, выявляют следующие требования к криптосистемам:

1. Вычислительно легко создавать пару открытый ключ (K_o), закрытый ключ (K_c).

2. Вычислительно легко, имея открытый ключ и незашифрованное сообщение M, создать соответствующее зашифрованное сообщение: $C = E_{Ko}[M]$.

3. Вычислительно легко дешифровать сообщение, используя закрытый ключ: $M = D_{Kc}[C] = D_{Kc}[E_{Ko}[M]]$.

4. Вычислительно невозможно, зная открытый ключ K_o, определить закрытый ключ K_c.

5. Вычислительно невозможно, зная открытый ключ K_o и зашифрованное сообщение C, восстановить исходное сообщение M.

Можно добавить шестое требование, хотя оно не выполняется для всех алгоритмов с открытым ключом:

6. Шифрующие и дешифрующие функции могут применяться в любом порядке, т.е. $M = E_{Ko}[D_{Kc}[M]] = D_{Kc}[E_{Ko}[M]]$.

Таким образом, данные, зашифрованные открытым ключом, можно расшифровать только секретным ключом. Следовательно, открытый ключ может распространяться через обычные коммуникационные сети и другие открытые каналы, что устраняет главный недостаток стандартных криптографических алгоритмов: необходимость использовать специальные каналы связи для распределения ключей.

3.2.1. Криптосистема RSA

В настоящее время лучшим и наиболее популярным криптографическим алгоритмом с открытым ключом считается RSA,

название которого получено по именам его создателей: Rivest, Shamir и Adelman.

Наиболее важной частью алгоритма RSA, как и других алгоритмов с открытым ключом, является процесс создания пары открытый/секретный ключи. В RSA он состоит из следующих шагов:

1. Случайным образом выбираются два секретных простых числа p и q таких, что $p \approx q$.

2. Вычисляется их произведение $r = p*q$.

3. Вычисляется функция Эйлера $\varphi(x)$ для $x=r$. Функция Эйлера $\varphi(x)$ для произвольного x представляет собой число взаимно простых с x чисел меньших, чем x. Для простого x $\varphi(x) = x–1$, а для числа представляющего собой произведение двух простых чисел $x = p*q$: $\varphi(x) = (p–1)*(q–1)$.

4. Выбирается целое значение открытой экспоненты e такой, что $1<e<\varphi(r)$ и $(e, \varphi(r)) = 1$.

5. Вычисляется значение секретной экспоненты d, которая должна удовлетворять условию $(e*d) \bmod \varphi(r) = 1$ (т.е., d является мультипликативной инверсной по модулю $\varphi(r)$ для e).

Таким образом, ключом шифрования K_o является пара значений (e, r), а ключом дешифрования K_c – (d, r). Значение параметра r, так же как и значение e являются общедоступной информацией, в то время как значения параметров p, q, d и $\varphi(r)$ хранятся в секрете.

Перед шифрованием исходное сообщения M необходимо разбить на блоки $M = m_1,m_2,m_3,\ldots$, где m_i представляет собой число в диапазоне от 0 до $r–1$. Сам процесс шифрования открытым ключом $K_o = (e, r)$ последовательности чисел m_i происходит согласно формуле:

$$c_i = (m_i{}^e) \bmod r, \qquad (3.1)$$

где последовательность чисел c_i представляет собой шифротекст.

Чтобы расшифровать эти данные секретным ключом $K_c = (d, r)$, необходимо выполнить следующие вычисления:

$$m_i = (c_i{}^d) \bmod r. \qquad (3.2)$$

В результате будет получено множество чисел m_i, которые представляют собой исходный текст.

Приведем простой пример использования метода RSA для шифрования сообщения "CAT". Сначала получим ключи:

1. Выберем $p = 3$, $q = 11$.

2. Вычислим $r = 3*11 = 33$.

3. Вычислим $\varphi(r) = (p–1)*(q–1) = 2*10 = 20$.

4. Выберем открытую экспоненту e, которая является взаимно простой с $\varphi(r) = 20$, например, $e = 7$.

5. На основе e и $\varphi(r)$ вычислим закрытую экспоненту d. Для этого можно использовать расширение алгоритма Евклида:

```
EUCLIDEX(a; b)
d0:=a; d1:=b;
x0:=1; x1:=0;
y0:=0; y1:=1;
while d1>1 do
begin
q:=d0 div d1;
d2:=d0 mod d1;
x2:=x0-q*x1;
y2:=y0-q*y1;
d0:=d1; d1:=d2;
x0:=x1; x1:=x2;
y0:=y1; y1:=y2;
end
return (x1; y1; d1)
```

Расширенный алгоритм Евклида позволяет вычислить числа x_1 и y_1, для которых выполняется равенство $x_1*a+y_1*b = d_1$, где d_1 = НОД(a, b). Если a и b взаимно простые, и $a>b$, то y_1 является мультипликативным инверсным по модулю a для b, т. е. $y_1*b \bmod a = 1$. Используя данный алгоритм, можно вычислим d, положив a, равным $\varphi(r)$, и b, равным $e = 7$. Если значение y_1 получилось отрицательным, то для получения корректного значения d необходимо добавить к y_1 значение a (или $\varphi(r)$).

В нашем случае имеем $d = y_1 = 3$.

6. Представим шифруемое сообщение как последовательность целых чисел в диапазоне 2...27 (для английского языка). Пусть букве 'A' соответствует число 2, 'B' – 3, 'C' – 4 и так далее. Тогда сообщение M = "CAT" можно представить в виде последовательности чисел $m_1,m_2,m_3 = \{4, 2, 21\}$. Зашифруем сообщение, используя открытый ключ $K_o = (7, 33)$:

$c_1 = (m_1^e) \bmod r = (4^7) \bmod 33 = 16384 \bmod 33 = 16,$
$c_2 = (m_2^e) \bmod r = (2^7) \bmod 33 = 128 \bmod 33 = 29,$
$c_3 = (m_3^e) \bmod r = (21^7) \bmod 33 = 1801088541 \bmod 33 = 21.$

7. Для расшифровки полученного сообщения $C = \{16, 29, 21\}$ с помощью секретного ключа $K_c = (3, 33)$ необходимо выполнить действия:

$m_1 = (c_1^d) \bmod r = (16^3) \bmod 33 = 4096 \bmod 33 = 4,$
$m_2 = (c_2^d) \bmod r = (29^3) \bmod 33 = 24389 \bmod 33 = 2,$
$m_3 = (c_3^d) \bmod r = (21^3) \bmod 33 = 9261 \bmod 33 = 21.$

Таким образом, в результате расшифровки сообщения получено исходное сообщение {4, 2, 21} = "CAT".

Для возведения в степень $x = a^z \bmod n$ можно использовать алгоритм быстрого возведения в степень по модулю:

```
function fast_exp(a,z,n)
begin
a1:=a
z1:=z
x:=1
while z1<>0 do
begin
while (z1 mod 2)=0 do
begin
z1:=z1 div 2
a1:=(a1*a1) mod n
```

```
end
z1:=z1-1
x:=(x*a1) mod n;
end
fast_exp:=x
end
```

Криптостойкость алгоритма RSA основывается на двух математических трудно решаемых задачах, для которых не существует эффективного способа их решения. Первая из них заключается в том, что невозможно вычислить исходный текст из шифротекса, так как для этого надо извлечь корень степени e по модулю числа r (найти дискретный логарифм). Данную задачу в настоящее время невозможно решить за полиномиальное время. С другой стороны практически невозможно найти секретный ключ, зная открытый, поскольку для этого необходимо решить сравнение $e*d \equiv 1 \pmod{\varphi(r)}$. Для его решения нужно знать делители целого числа r, т.е. разложить число r на сомножители. Задача разложения на множители, задача факторизации числа, в настоящее время также не имеет эффективного (полиномиального) решения, но пока не было доказано, что эффективного алгоритма решения данной задачи не существует.

На практике же применяются 1024–2048-битные числа r, однако предсказывается, что в скором времени 1024-битные ключи будут взломаны, а 2048-битные будут криптостойкими до 2030. Поэтому лучше использовать 3072-битные ключи, если необходимо обеспечить секретность вплоть до 2030г. Так, если взять ключ r длиной 300 бит, то разложить на множитель его можно за несколько часов на обычном персональном компьютере.

3.2.2. Криптосистема Эль-Гамаля

Данная криптосистема, предложенная в 1984 году, лежит в основе стандартов электронной цифровой подписи в США и России.

Как и для алгоритма RSA, для криптосистемы Эль-Гамаля необходимо перед шифрованием вычислить пару открытый/закрытый ключ. Это происходит по следующему алгоритму:

1. Генерируется случайное простое число p.

2. Выбирается произвольное целое число g, являющееся первообразным корнем по модулю p. Первообразным корнем по модулю некоторого числа p является целое число g такое, что $g^{\varphi(p)} = 1 \bmod p$ и $g^l \neq 1 \bmod p$ при $1 \le l \le \varphi(p)-1$.

3. Выбирается случайное целое положительное число $x < p-1$, не равное 1.

4. Вычисляется $y = g^x \bmod p$.

Открытым ключом K_o является тройка (p, g, y), закрытым ключом K_c – число x.

Сообщение $m \in [0, p-1]$ шифруется так:

1. Выбирается случайное секретное число $k \in (1, p-1)$, взаимно простое с $p-1$.

2. Вычисляется два значения a и b:

$$a = g^k \bmod p,$$
$$b = y^k m \bmod p, \quad (3.3)$$

где m является исходным сообщением, а пара чисел (a, b) – шифротекстом.

Как видно, полученный шифротекст в два раза длиннее исходного сообщения. Также следует отметить, что для разных сообщений m_1 и m_2 следует использовать и разные значения случайного k. Например, если для двух текстов m_1 и m_2 было использовано одно и то же значение k, то получим, что $\frac{b_1}{b_2} = \frac{m_1}{m_2}$, и значение m_2 может быть легко вычислено злоумышленником, при известном m_1.

Зная закрытый ключ x, исходное сообщение можно вычислить из шифротекста (a, b) по формуле:

$$m = ba^{-x} \bmod p \quad (3.4)$$

или по формуле:

$$m = ba^{(p-1-x)} \bmod p \quad (3.5)$$

Приведем пример шифрования сообщения $m = 4$ с помощью алгоритма Эль-Гамаля. Для начала сгенерируем ключи:

1. Выберем простое число $p = 13$.

2. Подберем для него такое g, которое являлось бы первообразным корнем по модулю p. Для этого используем алгоритм нахождения случайного первообразного корня по модулю p:

```
in: p, q1, q2, ..., qk // q1,q2,...,qk – все простые делители числа p-1
out: g
Root (p, q1, q2, ..., qk)
begin
g:=random(p-1); // генерируем случайное число из диапазона [2, p-1]
for (i:=1; i<k+1; i++)
      if ( exp(g,(p-1)/qi)== 1 (mod p)) return Root (p, q1, q2, ..., qk);
return g;
end
```

Пусть $g = 7$.

При $p = 13$, $\varphi(13) = 13 - 1 = 12$.

$g^{\varphi(p)} = 1 \bmod p = 7^{12} \bmod 13 = 1$

$7^1 \ mod\ 13 = 7 \quad (\neq 1);$
$7^2 \ mod\ 13 = 10 \quad (\neq 1);$
$7^3 \ mod\ 13 = 5 \quad (\neq 1);$
…
$7^{11} \ mod\ 13 = 2 \quad (\neq 1);$

По результатам проверки принимаем решение об использовании g равного 7.

3. Закрытым ключом x возьмем число 5.

4. Вычисляем $y = g^x \bmod p = 7^5 \bmod 13 = 11$.

Открытым ключом является K_o = (13, 7, 11), закрытым ключом K_c – число 5.

Далее зашифруем сообщение m = 4, для чего сгенерируем случайное число k = 7, взаимно простое с p−1, и вычислим по формуле 3.3 значения a и b:

$a = g^k \bmod p = 7^7 \bmod 13 = 6,$
$b = y^k m \bmod p = 11^7 * 4 \bmod 13 = 8.$

На выходе получаем шифротекст (6, 8).

Вычислим исходное сообщение из шифротекста (a, b) по формуле 3.4:

$$m = b * a^{-x} \bmod p = b * (a^x)^{-1} \bmod p = b * (a^x)^{\varphi(p)-1} \bmod p$$
$$= 8 * (6^5)^{11} \bmod 13 = 8 * (7776)^{11} \bmod 13 = 8 * 2^{11} \bmod 13$$
$$= 16384 \bmod 13 = 4$$

Криптостойкость криптосистемы Эль-Гамаля основана на том, что невозможно вычислить закрытый ключ, зная открытый, за полиномиальное время, так как для этого необходимо вычислить дискретный логарифм, т.е. по известным значениям p, g и y вычислить x, который бы удовлетворял сравнению: $y \equiv g^x \bmod p$.

3.2.3. Криптосистема Рабина

Криптосистема, разработанная Рабином, подобно RSA основывается на трудной проблеме факторизации больших целых чисел или на проблеме извлечения квадратного корня по модулю составного числа $n = p*q$. Эти две задачи являются эквивалентными, т. е. зная простые делители числа n, можно извлечь квадратные корни по модулю n, а умея извлекать квадратные корни по модулю n, – разложить n на простые множители.

Генерация ключей в криптосистеме Рабина происходит следующим образом:

1. Выбираются два разных случайных простых числа p и q таких, что $p \approx q$. При этом они должны удовлетворять условию: $p \equiv q \equiv 3 \bmod 4$.

Выполнение данного условия необходимо для упрощения вычисления квадратного корня по модулю p и q при дешифрации сообщения.

2. Вычисляется $n = p*q$.

Открытым ключом будет значение n, в то время как простые числа p и q – закрытым.

Для получения шифротекста c_i необходимо выполнить следующие действия над исходным сообщением M, которое также разбивается на блоки $m_1m_2m_3..$, $(0 \le m_i \le n-1)$:

$$c_i = m_i^2 \bmod n. \tag{3.6}$$

Процесс шифрования в алгоритме Рабина происходит намного быстрее, чем в других криптосистемах с открытым ключом.

Для дешифрации же нужно решить квадратное уравнение вида $m_i^2 + b*m_i - c_i = 0 \pmod n$. Как известно, общее решение такого уравнения будет иметь вид:

$$m = \frac{-b + \sqrt{D}}{2} (\bmod n),$$

где $D = b^2+4c \pmod n$.

Вычислить квадратный корень из D по модулю числа $n=p*q$ можно с помощью китайской теоремы об остатках, для чего необходимо знать закрытые ключи p и q (3.7). Полученные результаты подставляем в (3.8) и вычисляем значения m_1, m_2, m_3, m_4. Одно из полученных значений m_i и является исходным текстом.

Обобщенный алгоритм расшифрования выглядит следующим образом:

1. Вычисляем параметры m_p и m_q с помощью китайской теоремы об остатках:

$$m_p = \sqrt{D}\ mod\ p = c^{\frac{p+1}{4}}\ mod\ p$$
$$m_q = \sqrt{D} \bmod q = c^{\frac{q+1}{4}} \bmod q \tag{3.7}$$

   где c – шифротекст, p и q – закрытый ключ

2. Вычисляем значения y_p и y_q таких, что $y_p*p+y_q*q = 1$ по расширенному алгоритму Евклида.

3. Вычисляем значения m_1, m_2, m_3, m_4 по модулю n по формулам:

$$m_1 = (y_p * p * m_q + y_q * q * m_p) \bmod n$$
$$m_2 = n - m_1,$$
$$m_3 = (y_p * p * m_q - y_q * q * m_p) \bmod n, \tag{3.8}$$
$$m_4 = n - m_3.$$

Главным недостатком криптосистемы Рабина является то, что неизвестно который из четырех результатов, m_1, m_2, m_3, m_4 равен исходному m_i. Если сообщение написано по-русски, выбрать правильное m_i, нетрудно. С другой стороны, если сообщение является потоком случайных битов, способа определить, какое m_i, – правильное, нет. Одним из способов решить эту проблему служит добавление к сообщению перед шифрованием известного заголовка.

Приведем пример работы данного алгоритма. Пусть нам надо зашифровать сообщение 'P'. 'P' является 17-ой буквой алфавита, поэтому m = 17. Сначала сгенерируем ключи:

1. Выберем два простых числа p = 11 и q = 19. При этом 11 mod 4 = 3 и 19 mod 4 = 3. Т.е. условие $p \equiv q \equiv 3 \bmod 4$ выполняется.

2. Вычислим наш открытый ключ: $n = p*q = 11*19 = 209$.

Далее зашифруем сообщение m = 17. Для этого вычислим по формуле 3.6 значение $c = m^2 \bmod n = 17^*17 \bmod 209 = 80$. Таким образом, получили шифротекст c = 80.

Для расшифровки сообщения выполняем следующие действия:

1. Вычислим m_p и m_q по формуле:

$$m_p = \sqrt{D} \bmod p = c^{\frac{p+1}{4}} \bmod p = 80^{\frac{11+1}{4}} \bmod 11 = 80^3 \bmod 11 = 5,$$

$$m_q = \sqrt{D} \bmod q = c^{\frac{q+1}{4}} \bmod q = 80^{\frac{19+1}{4}} \bmod 19 = 80^5 \bmod 19 = 17.$$

2. Вычислим значения y_p и y_q таких, что $y_p*p+y_q*q = 1$ по расширенному алгоритму Евклида. Для этого принимаем a = 11, b = 19. Тогда на выходе получим, что $y_p = x_1 = 7$, а $y_q = y_1 = -4$.

3. Далее вычисляем значения m_1, m_2, m_3, m_4 по модулю n по формулам:

$m_1 = (y_p * p * m_q + y_q * q * m_p) \bmod n = (7*11*17 + (-4)*19*5) \bmod 209 = (1309 - 380) \bmod 209 = 929 \bmod 209 = 93$,

$m_2 = n - m_1 = 209 - 93 = 116$,

$m_3 = (y_p * p * m_q - y_q * q * m_p) \bmod n = (7*11*17 - (-4)*19*5) \bmod 209 = (1309 + 380) \bmod 209 = 1689 \bmod 209 = 17$,

$m_4 = n - m_3 = 209 - 17 = 192$.

В результате расшифровки получаем: $m \in \{93, 116, \mathbf{17}, 192\}$. Видим, что один из корней является исходным текстом m.

3.3 ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Изучить теоретический материал по лабораторной работе.
2. Выполнить задание согласно своему варианту (см. табл. 3.1).

3. Реализованное программное средство должно зашифовывать и расшифровывать произвольный текстовый файл (*.txt), размером более 1kb.
4. Длины ключей во всех алгоритмах должны быть достаточно большими (512 бит - 1024 бита). Можно использовать для работы классы работы с большими числами (BigInteger C#, Java).

Пояснение по выбору варианта:

Ваш_Номер_Варианта=(Номер_Вашей_зачетной_книжки) mod 3+1

Например №=123456

Выполнить задание согласно своему варианту (см. табл. 3.1).

(123456) mod3 +1 = 0+1=1

Таблица 3.1.

Вариант	Задание
№1	Реализовать шифратор и дешифратор алгоритма RSA, используя алгоритм быстрого возведения в степень. А также реализовать вычисление открытого ключа K_o при данных значениях p, q и e, используя расширенный алгоритм Евклида.
№2	Реализовать шифратор и дешифратор алгоритма Эль-Гамаля, используя алгоритм быстрого возведения в степень. А также реализовать вычисление открытого ключа g при данном значении p, используя алгоритм нахождения первообразного корня по модулю.
№3	Реализовать шифратор и дешифратор по алгоритму Рабина, используя расширенный алгоритм Евклида и алгоритм быстрого возведения в степень при дешифрации.