Министерство образования и науки Российской Федерации
Сибирский федеральный университет

ОСНОВЫ ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ

Учебно-методическое пособие
(ПРОЕКТ)

Красноярск
СФУ
2016

ЛАБОРАТОРНАЯ РАБОТА №2
(Основы криптографии с открытым ключом. Алгоритм RSA)

Цель работы:

- ознакомиться с основами ассиметричной криптографии;
- ознакомиться с элементами теории чисел, используемых в криптографии с открытым ключом;
- изучить особенности алгоритма с открытым ключом RSA;
- получить навыки разработки криптосистем с открытым ключом с использованием языка программирования высокого уровня;

1. ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Асимметричные алгоритмы шифрования называются также алгоритмами с открытым ключом. В отличие от алгоритмов симметричного шифрования (алгоритмов шифрования с закрытым ключом), в которых для шифрования и расшифрования используется один и тот же ключ, в ассиметричных алгоритмах один ключ используется для шифрования, а другой, отличный от первого, – для расшифрования. Алгоритмы называются асимметричными, так как ключи шифрования и расшифрования разные, следовательно, отсутствует симметрия основных криптографических процессов (см. рис .1). Один из двух ключей является открытым (*public key*) и может быть объявлен всем, а второй – закрытым (*private key*) и должен держаться в секрете. Какой из ключей, открытый или закрытый, используется для шифрования, а какой для расшифрования, определяется назначением криптографической системы.

В настоящее время асимметричные алгоритмы широко применяются на практике, например, в протоколах выработки общего секретного ключа, в протоколах электронной подписи, в протоколах безопасной передачи информации в сети интернет и др.

Алгоритмы шифрования с открытым ключом можно использовать для решения, как минимум, трех задач:

- для шифрования передаваемых и хранимых данных в целях их защиты от несанкционированного доступа;
- для формирования цифровой подписи электронных документов;
- в задачах распределения секретных ключей, используемых в дальнейшем при симметричном шифровании.

1.1. Односторонние функции

Стойкость асимметричной криптосистемы обеспечивается за счет особых свойств криптографических преобразований, которые в комплексе представляет так называемую одностороннюю функцию. Односторонней функцией (*one-way function*) называют такую математическую функцию, которую относительно легко вычислить, но трудно найти по значению функции соответствующее значение аргумента. То есть, зная x легко вычислить $f(x)$, но по известному $f(x)$ трудно вычислить справедливое значение x. Причём следует отметить, что под понятием «трудно вычислить» понимают, что для этого потребуется не один год расчетов с использованием современных ЭВМ. Односторонние функции применяются в криптографии также в качестве хеш-функций. Использовать односторонние функции для шифрования сообщений с целью их защиты не имеет смысла, так как обратно расшифровать зашифрованное сообщение уже не получится, т.е. говорят, что преобразования односторонней функции необратимы. Для целей шифрования используются специальные однонаправленные функции – однонаправленные функции с «секретом», особый вид односторонних функций, имеющих некоторый «секрет», позволяющий относительно быстро вычислить обратное значение функции. Строго говоря, не доказано, что однонаправленные функции существуют. Однако признано, что некоторые преобразования обладают свойствами, близкими к свойствам однонаправленных функций. Они широко используются в действующих системах криптографической защиты информации.

Для однонаправленной функции $f(x)$ с «секретом» справедливы следующие утверждения:

- зная x, легко вычислить $f(x)$,
- по известному значению $f(x)$ трудно найти соответствующее x,
- зная дополнительно некоторую секретную информацию, можно легко вычислить x.

1.2. Использование ассиметричных алгоритмов в криптографии

В 1976 году Уитфилд Диффи и Мартин Хеллман предложили принцип шифрования, основанный на использовании двух разных ключей, хотя и связанных между собой, но устроенных таким образом, что вычислить по одному из них (открытому) другой (закрытый) практически невозможно. Этот алгоритм получил название «Алгоритм Диффи-Хелмана» (англ. *Diffie-Hellman*, DH).

Суть алгоритма Диффи-Хелмана заключается в следующем. Предварительно распределяемые закрытые ключи вообще не должны использоваться для шифрования данных (так как секрет, который известен более чем одному человеку, – уже не секрет). Закрытый ключ должен быть известен только одному лицу – его владельцу. Такой принцип использования асимметричных алгоритмов получил название открытого шифрования или шифрованием с открытым ключом.

Согласно этому принципу, любой желающий может зашифровать сообщение открытым ключом. Расшифровать сообщение сможет только владелец закрытого ключа. Пусть, например, существуют два пользователя сети Алиса и Боб, которые могут обмениваться сообщениями по открытому каналу связи Алиса и Боб. Для обмена сообщениями они используют описанный выше способ открытого шифрования. Предположим, что Алисе необходимо передать секретное сообщение Бобу так, чтобы никто другой не смог его прочитать. Для этого необходимо выполнить следующие действия:

1. Боб посылает Алисе свой открытый ключ O по любому каналу связи, например, по электронной почте.
2. Алиса шифрует свое сообщение M полученным открытым ключом O и получает зашифрованное сообщение C.
3. Зашифрованное сообщение C пересылается Бобу по тому же каналу связи.
4. Боб расшифровывает полученное сообщение C своим закрытым ключом K.

Обозначим операцию шифрования как E, а операцию расшифровывания как E^{-1}, тогда схема протокола обмена ключами можно представить на рис. 2.

Ассиметричная криптография породила еще одну громадную отрасль современной криптографии – ***электронную подпись*** **(*ЭП*)**. При формировании электронной подписи под сообщением, отправляемым неким абонентом в той же информационной системе, отправитель *подписывает* послание своим секретным ключом. На самом деле пользователь вычисляет контрольную сумму сообщения, шифрует ее секретным ключом и присоединяет шифрограмму к сообщению, однако глагол «подписывает» очень удачно вписался в новое значение и теперь неотделим от понятия ЭП.

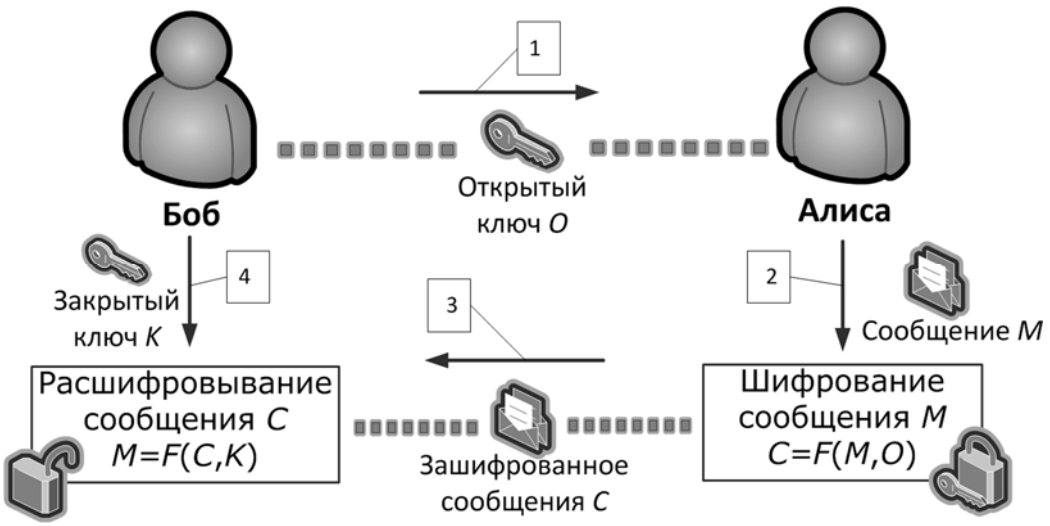

Рис. 1. Схема обмена ключами Диффи-Хелмана

1.3. Электронная подпись (ЭП)

При создании электронной подписи под документом в нее закладывается достаточно информации чтобы любой получатель мог удостовериться с помощью открытого ключа отправителя, что только он мог подписать, а, следовательно, и отправить это сообщение. Но при этом в подписи не должно быть достаточно информации, чтобы извлечь из нее сам секретный ключ отправителя – иначе после первого подписания любой перехвативший письмо мог бы подписывать свои послания, т. е. технология ЭП очень напоминает асимметричный шифр, только наоборот (для шифрования задействован закрытый ключ).

Следует отметить, что асимметричное шифрование и ЭП решают совершенно различные задачи: первое – обеспечение конфиденциальности послания, второе – аутентичность (достоверность) отправителя и целостность сообщения.

Основу ЭП (как разновидности ассиметричной криптографии) также составляют однонаправленные функции, а также хэш-функции. В качестве открытого ключа выбирается какое-либо частное уравнение, которое и является этой трудноразрешимой задачей. Но при составлении этого уравнения оно разрабатывалось так, что лицо, знающее некоторую дополнительную информацию («секрет») об этом уравнении, может решить его за разумный временной интервал. Эта дополнительная информация и является закрытым ключом.

При формировании ЭЦП процесс идет в следующем порядке. Отправитель письма добавляет к письму некоторую часть закрытого ключа в таком виде, чтобы по ней невозможно было полностью восстановить закрытый ключ. Однако, этой информации достаточно, чтобы помочь любому желающему (проверяющему ЭП) решить то самое уравнение, на базе которого построена данная схема ЭП. В качестве других параметров уравнения проверяющий подставляет контрольную

сумму полученного письма и значения из открытого ключа отправителя. Если уравнение успешно разрешилось, т. е. при подстановке всех данных оно превратилось в равенство, значит, письмо с данной контрольной суммой отправил именно тот абонент, чья подпись под ним стоит. Если же равенство не получилось, то на каком–то из этапов произошел сбой и необходимо уже более тщательно выяснить, был ли это случайный сбой на канале связи или же было умышленное вмешательство.

В асимметричной криптографии и ЭП ключ является более сложным компонентом, чем просто 128-битный блок данных симметричной криптографии. Чаще всего и открытый и закрытый ключи состоят из двух или трех очень больших (сотни и даже тысячи бит) натуральных чисел, однако встречаются и случаи, когда ключ состоит из сотен натуральных чисел – определенной последовательности или двумерной матрицы. Поэтому на практике в качестве ключа, или электронной подписи может использоваться наборы чисел – все это неразделимый ключ, имеющий смысл только в сочетании всех ее компонент. Также следует отметить, что размерность ключей может достигать сотен килобайт, что для ассиметрично криптографии является нормальным показателем

1.4. Алгоритм шифрования с открытым ключом RSA

Алгоритм шифрования с открытым ключом RSA был предложен в 1977 году. Его название составлено из первых букв фамилий авторов: Р.Райвеста (R.Rivest), А.Шамира (A.Shamir) и Л.Адлемана (L.Adleman). Алгоритм RSA является очень популярным и широко применяемым асимметричным алгоритмом в криптографических системах.

Алгоритм основан на использовании того факта, что задача разложения большого числа на простые сомножители является трудной. Криптографическая система RSA базируется на следующих двух фактах из теории чисел:

- задача проверки числа на простоту является сравнительно легкой;
- задача разложения чисел вида $n = p \cdot q$ (где p и q – простые числа) на множители является очень трудной, если мы знаем только n, а p и q – большие числа (задача факторизации больших чисел).

Алгоритм RSA представляет собой блочный алгоритм шифрования, где зашифрованные и незашифрованные данные должны быть представлены в виде целых чисел в диапазоне $(0, N-1)$ для некоторого N.

1.4.1. Подготовка к процессу шифрования

Пусть абонент А хочет передать зашифрованное сообщение абоненту Б. В этом случае абонент Б должен подготовить пару ключей (открытый и закрытый и отправить свой открытый ключ пользователю А.

Первым этапом является генерация открытого и закрытого ключей. Для этого последовательно проделываются следующие действия:

1. выбираются два *больших* и *простых* числа P и Q;
2. вычисляется произведение $N = P \cdot Q$;
3. вычисляется значение функции Эйлера $\varphi(N) = (P-1)(Q-1) = d$;
4. случайным образом выбирается число $s < d$ и взаимно простое с d;
5. вычисляется число e, такое, что $e \cdot s \mod d = 1$.

После всех этих операций, числа s и N будут открытым ключом пользователя, а число e – закрытым ключом.

Так как пользователь Б хочет получить зашифрованное сообщение от пользователя А, значит пользователь Б должен отправить свой открытый ключ (s, N) пользователю А. Числа P и Q больше не нужны, но при этом они остаются уязвимым местом алгоритма и поэтому подлежат удалению.

1.4.2. Процесс шифрования и дешифрования

Абонент А представляет своё сообщение в цифровом виде (используя, например, одну из кодировок символов) и разбивает его на блоки $m_1, m_2, ...m_i$ где $m_i < N$. Соответственно, после шифрования, зашифрованное сообщение будет состоять из блоков c_i.

Далее абонент А шифрует каждый блок своего сообщения по формуле $c_i = m_i^s \mod N$. После этого зашифрованное сообщение $C = \langle c_1, c_2, ..., c_i \rangle$ пересылается по открытой линии.

Абонент Б, получивший зашифрованное сообщение, расшифровывает все блоки полученного сообщения по формуле $m_i = c_i^e \mod N$. В случае, если все зашифрованные блоки были переданы без ошибок (случайных или преднамеренных) все расшифрованные блоки будут точно такими же, как и исходящие от пользователя А.

Злоумышленник, перехватывающий все сообщения и знающий всю открытую информацию, не сможет найти исходное сообщение при больших значениях P и Q.

1.4.3. Пример использования алгоритма

Пусть пользователь А хочет передать пользователю Б сообщение. В этом случае вначале пользователь Б должен подготовить открытый и закрытый ключи. Пусть, например, им выбраны следующие параметры:

$$P = 7,\ Q = 13, \text{ соответственно } N = P \cdot Q = 7 \cdot 13 = 91.$$

Затем пользователь Б вычисляет значение функции Эйлера $\varphi(N) = d = (P-1)(Q-1) = 72$ и любое число *s*, не имеющее общих делителей с *d* (это необходимо для того, чтобы зашифрованное сообщение можно было потом однозначно восстановить). Пусть *s* = 11. Это число будет одним из компонентов открытого ключа.

Далее необходимо найти число *e*, которое можно будет использовать в качестве закрытого ключа для расшифрования сообщения. Значение *e* должно удовлетворять соотношению $e \cdot s \mod d = 1$. Для малых значений *d* число *e* можно найти подбором, однако в общем случае для поиска *e* можно использовать *обобщенный алгоритм Евклида*. Для нашего случая нам необходимо найти такое число *e*, что $e \cdot 11 \mod 72 = 1$. Методом перебора достаточно быстро убедиться, что число *e*=59 ($59 \cdot 11 \mod 72 = 1$).

Теперь пользователь Б запоминает свой закрытый ключ *e*=59 и отправляет открытый ключ (11, 91) пользователю А и уничтожает числа *P* = 7 и *Q* = 13. Пользователь А, получивший открытый ключ (11, 91), увидев, что *N*=91, разбивает исходное сообщение, например, на три блока, причем значение каждого меньше *N*. Предположим, что в результате разбиения имеется три блока $m_1 = 8$, $m_2 = 27$ и $m_3 = 5$. Затем пользователь А шифрует каждый блок:

$$c_1 = 8^{11} \mod 91 = 57$$
$$c_2 = 27^{11} \mod 91 = 27.$$
$$c_3 = 5^{11} \mod 91 = 73$$

Зашифрованное сообщение, состоящее из трех блоков (57, 27, 73), передается пользователю Б, который, используя свой закрытый ключ *e* = 59 и *N*=91, расшифровывает сообщение:

$$m_1 = 57^{59} \mod 91 = 8$$
$$m_2 = 27^{59} \mod 91 = 27.$$
$$m_3 = 73^{59} \mod 91 = 5$$

Таким образом, абонент Б расшифровал сообщение от абонента А.

1.4.4. Особенности вычислений

Как шифрование, так и расшифрование в алгоритме RSA основывается на использовании операции возведения целого числа в целую степень по некоторому модулю *N*. Однако даже на приведённом выше элементарном примере легко убедиться, что если сначала возводить в степень и только потом проводить сравнение по модулю *N*, промежуточные значения окажутся огромными. Чтобы избежать этого, можно воспользоваться свойствами арифметики остатков и теории сравнений. Для практической реализации данного алгоритма полезно напомнить следующие свойства арифметики остатков:

1. $a \equiv b \pmod N$, для двух чисел *a* и *b*, если они имеют одинаковые остатки от деления на *m*. Аналогично это свойство можно записать и так: $a \mod N = b$;
2. если $a - b$ делится на *N*, то $a \equiv b \pmod N$, например $15 \equiv 1 \pmod 7$, т.к. $15 - 1 = 14$, а 14 кратно 7 (делится без остатка).
3. если $a \equiv b \pmod N$, то $a^k \equiv b^k \pmod N$, где $k \subset \mathrm{N}$.
4. если $ac \equiv bc \pmod N$ и *c* взаимно простое с *N*, то $a \equiv b \pmod N$.
5. если $ac \equiv bc \pmod{Nc}$, то $a \equiv b \pmod N$. Например $44 \equiv 4 \pmod{40}$, что тоже самое, что и $11 \cdot 4 \equiv 1 \cdot 4 \pmod{10 \cdot 4}$, а значит $11 \equiv 1 \pmod{10}$;
6. Теорема Ферма-Эйлера (для случая алгоритма RSA). Если *p* и *q* – два различных простых числа, а *m* – любой число, которое не делится на *p* и *q*, то $m^{(p-1)(q-1)} \equiv 1 \pmod{p \cdot q}$. Например, пусть *p*=11, *q*=5, $p \cdot q = 55$ и *m*=3. Убедимся, что $3^{40} \equiv 1 \pmod{55}$.

$$3^{40} \bmod 55 = 3^{(11-1)(5-1)} \bmod 55 = \left(3^{10}\right)^4 \bmod 55 = 59049^4 \bmod 55 = 34^4 \bmod 55 = 1.$$

Здесь необходимо заметить, что предпоследнее равенство получено на основании свойства 3.

7. Для любых *a* и *b* и выполняется равенство: $(ab) \bmod N = \left(\left(a \bmod N\right) \cdot \left(b \bmod N\right)\right) \bmod N$.

1.4.5. Применение алгоритма RSA на практике

На протяжении многих лет алгоритм RSA активно используется как в виде самостоятельных криптографических продуктов, так и в качестве встроенных средств в популярных приложениях. Открытое шифрование на базе алгоритма RSA применяется в популярном пакете шифрования PGP, операционной системе *Windows*, различных Интернет-браузерах, банковских компьютерных системах. Кроме того, различные международные стандарты шифрования с открытым

ключом и формирования цифровой подписи используют RSA в качестве основного алгоритма.

Для обеспечения высокой надежности шифрования необходимо, чтобы выступающее в качестве модуля число N было очень большим – несколько сотен или тысяч бит. Только в этом случае будет практически невозможно по открытым параметрам определить закрытый ключ. Современное состояние технических средств разложения на множители таково, что число, содержащее 193 десятичных знака, факторизовано в 2005 г. Следовательно, выбираемое число N должно быть больше.

Сами авторы *RSA* рекомендовали использовать следующие размеры модуля N: 768 бит - для частных лиц; 1024 бит - для коммерческой информации; 2048 бит - для особо секретной информации. С момента получения их рекомендаций прошло время, поэтому современные пользователи должны делать поправки в сторону увеличения размера ключей. Однако, чем больше размер ключей, тем медленнее работает алгоритм. Поэтому увеличивать размер ключа без необходимости не имеет смысла.

2. ЗАДАНИЯ К ЛАБОРАТОРНОЙ РАБОТЕ

Согласно Вашему персональному варианту (см. табл. 1) разработайте алгоритм шифрования/расшифровывания RSA, со следующими особенностями:

- объём исходного текста – любой (в разумных пределах);
- исходный текст может состоять из русских и английских букв, цифр, а также знаков препинания;
- исходный текст находится в кодировке ASCII (или Unicode);
- выступающее в качестве модуля число – N, которое выбирается автоматически и состоит из количества ДЕСЯТИЧНЫХ знаков, указанных в табл. 1. Числа P и Q выбираются случайным образом, так, что $P \cdot Q = N$, где P и Q – простые числа.
- исходный текст разбивается на K блоков, где K выбирается исходя из значения модуля N;

Убедитесь в правильности составления алгоритмов, а затем на языке программирования составьте программу, которая реализует данный алгоритм.

На ряде контрольных примеров (не менее 10) открытого текста проверьте правильность работы алгоритмов шифрования и дешифрования (в качестве контрольного примера понимается текстовый файл в кодировке ASCII).

Оцените криптостойкость вашего варианта алгоритма RSA, а также сделайте оценку производительности, разработанной Вами программы.

Разработанная Вами программа должна содержать графический интерфейс пользователя.

Таблица 1. Варианты заданий в лабораторной работе

№ вар.	Шифр	*N,* кол-во знаков
1	RSA	31
2		26
3		44
4		38
5		51
6		35
7		28
8		58
9		49
10		35
11		47
12		39
13		28
14		30
15		32
16		59
17		27
18		55
19		33
20		40

3. СОДЕРЖАНИЕ ОТЧЁТА ПО ЛАБОРАТОРНОЙ РАБОТЕ

После выполнения лабораторной работы и проверки адекватности полученных результатов, необходимо подготовить отчёт, включающий в себя:

- титульный лист;
- задание к лабораторной работе (согласно вашему варианту);
- описание алгоритма шифрования/расшифровывания (посредством блок-схемы, псевдокода, пошагово на естественном языке);
- оценка алгоритма шифрования (режимы работы, криптостойкость, производительность и т.п.);
- листинг составленной программы, реализующей алгоритмы (шрифт `Courier New`);

- контрольные примеры работы программы, с различными исходными текстами, ключами и/или другими параметрами (не менее 5);
- заключение, содержащее выводы о криптостойкости алгоритма RSA, в рамках Вашего варианта и производительности программы шифрования. Кроме этого выводы должны содержать описание полученных лично Вами знаний и навыков в ходе выполнения лабораторной работы.

Отчёт должен быть подготовлен в текстовом редакторе, согласно действующему стандарту организации на оформление студенческих работ.