# Дисциплина «Защита информации»

# Лабораторная работа № 3

**Ассиметричное шифрование**

# Теоретическое введение

### Математические основы асимметричного шифрования.

Основная идея асимметричного шифрования заключается в существовании сразу двух ключей для обмена информацией – открытого, известного любому желающему, и закрытого, который известен лишь получателю информации. Очевидно, что открытый и закрытый ключи генерируются одновременно и между ними существует определенная математическая связь. Основная задача проектировщика асимметричного алгоритма заключается в том, чтобы по известному открытому ключу было бы невозможно (очень трудоемко) получить секретный ключ шифрования. Для этого в основу асимметричных алгоритмов закладываются вычислительно трудные задачи факторизации, дискретного логарифмирования, проецирования точек на эллиптической кривой и т.д. Объединяет все эти задачи то, что они используют операцию получения остатка от целочисленного деления.

Для любого положительного целого числа *n* и любого *a* при делении *a* на *n* мы получаем некоторое целое частное *q* и остаток *r*, удовлетворяющий соотношению

*a* = *qn* + *r*, 0 ≤ *r* < *n*; *q* = int(*a/n*),

где int(x) обозначает наибольшее целое число, не превышающее x.

Если a является целым, а *n* - положительным, то *a* mod *n* определяется как остаток от деления *a* на *n*. Таким образом, для любого целого числа *a* можно записать

*a* = int(*a/n*) \* *n* + (*a* mod *n*).

Говорят, что два целых числа *a* и *b* являются сравнимыми по модулю *n*, если (*a* mod *n*) = (*b* mod *n*). Это записывается в виде: *a* ≡ *b* mod *n*.

Операции сравнения по модулю имеют следующие свойства:

1. *a* ≡ *b* mod *n*, *n* | (*a* - *b*) (*n* | *x* означает, что *n* делит *x* нацело).
2. Из (*a* mod *n*) = (*b* mod *n*) следует *a* ≡ *b* mod *n*.
3. Из *a* ≡ *b* mod *n* следует *b* ≡ *a* mod *n*.
4. Из *a* ≡ *b* mod *n* и *b* ≡ *c* mod *n* следует *a* ≡ *c* mod *n*.

Операции арифметики в классах вычетов обладают следующими свойствами:

1. [(*a* mod *n*) + (*b* mod *n*)] mod *n* = (*a* + *b*) mod *n*.
2. [(*a* mod *n*) - (*b* mod *n*)] mod *n* = (*a* - *b*) mod *n*.
3. [(*a* mod *n*) \* (*b* mod *n*)] mod *n* = (*a* \* *b*) mod *n*.

Пусть Zn обозначает множество всех не отрицательных целых чисел, которые меньше *n*:

Zn = {0, 1, 2, ..., (n - 1)}.

Это множество называется ещё множеством вычетов (остатков) по модулю *n*. Для арифметических операций по модулю *n* в этом множестве выполняются следующие свойства:

|  |  |
| --- | --- |
| **Свойство** | **Выражение** |
| Коммутативные законы | (*w* + *x*) mod *n* = (*x* + *w*) mod *n*, (*w* \* *x*) mod *n* = (*x* \* *w*) mod *n* |
| Ассоциативные законы | [(*w* + *x*) + *y*] mod *n* = [*w* + (*x* + *y*)] mod *n*, [(*w* \* *x*) \* *y*] mod *n* = [*w* \* (*x* \* *y*)] mod *n* |
| Дистрибутивный закон | [(*w* + *x*) \* *y*] mod *n* = [(*w* \* *y*) + (*x* \* *y*)] mod *n* |
| Тождества | (0 + *w*) mod *n* = *w* mod *n*, (1 \* *w*) mod *n* = *w* mod *n* |
| Аддитивный обратный (-*w*) | Для любого *w* ∈ Zn существует такое *z*, что *w* + *z* ≡ 0 mod *n* |

Существует одна особенность арифметики в классах вычетов, которая делает её отличной от обычной арифметики.Заметим сначала, что, как и в обычной арифметике, имеет место следующее свойство

если (*a* + *b*) ≡ (*a* + *c*) mod *n*, то *b* ≡ *c* mod *n*.

Данное свойство согласуется с существованием аддитивного обратного. Прибавив к обеим частям данного равенства аддитивное обратное элемента *а*, получим:

((-*a*) + *a* + *b*) ≡ ((-*a*) + *a* + *c*) mod *n*, *b* ≡ *c* mod *n*.

Однако следующее утверждение:

если (*a* \* *b*) ≡ (*a* \* *c*) mod *n*, то *b* ≡ *c* mod *n*

выполняется только при условии, что *a* и *n* взаимно просты (обозначается далее(*a*, *n*) = 1).

Если *p* является простым, то все элементы Zp будут взаимно простыми с *p*. Это даёт нам возможность добавить ещё одно свойство к тем, которые были приведены выше:

|  |  |
| --- | --- |
| **Свойство** | **Выражение** |
| Мультипликативный обратный (*w*-1) | Для любого *w* ∈ Zp существует *z*, что *w* \* *z* ≡ 1 mod *p* |

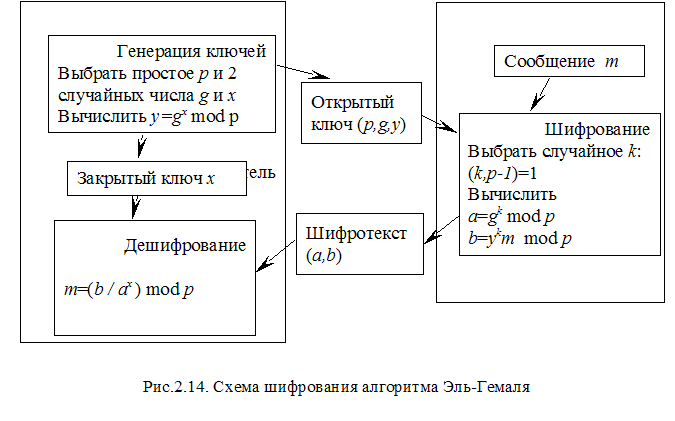
Для поиска мультипликативного обратного можно использовать расширенный алгоритм Евклида, который позволяет в целых числах найти решение уравнения *ax*+*by* = 1 при заданных *а* и *b*. Очевидно, что если решение существует, то *x* будет величиной, мультипликативно обратной *а* по модулю *b*.

Алгоритм Евклида

1. Определить матрицу *E*:  
   http://www.volpi.ru/umkd/zki/images/ImgFormula-2.5.1.gif
2. Вычислить *r* – остаток от деления числа *a* на *b*:  
   *a* = *bq* + *r*,0 ≤ *r* < *b*
3. Если *r* = 0, топервый столбец матрицы *E* является решением уравнения.
4. Если *r* ≠ 0, заменить матрицу *Е*:  
   http://www.volpi.ru/umkd/zki/images/ImgFormula-2.5.2.gif
5. Поменять местами столбцы матрицы *Е*.
6. Заменить пару чисел *a*, *b* на *b*, *r* и перейти к шагу 2.

### Алгоритмы асимметричного шифрования.

Исторически первой системой с открытым ключом стал *метод экспоненциального ключевого обмена Диффи* - Хеллмана, разработанный в 1976 году. Метод предназначен для передачи секретного ключа симметричного шифрования. В обмене задействованы два участника А и Б. Сначала они выбирают большие простые числа *n* и *g*<*n* (эти числа секретными не являются). Затем участник A выбирает большое целое число *х*,вычисляет *Х* = *gx* mod *n* и передает *Х* участнику Б. Б в свою очередь выбирает большое целое число *y*, вычисляет *Y* = *gy* mod *n* и передает *Y* участнику А. Б вычисляет *K’* = *Xy* mod *n*, А вычисляет *K’’*= *Yx* mod *n*. Легко заметить, что *K’* = *K’’* = *gxy* mod *n*, и это значение оба участника могут использовать в качестве ключа симметричного шифрования. Криптостойкость этого метода определяется трудоемкостью вычисления дискретного логарифма в конечном поле. Действительно, злоумышленник может узнать такие параметры алгоритма, как *n*, *g*, *X*, *Y*, но вычислить по ним значения *x* или *y* – задача, требующая очень больших вычислительных мощностей и времени. Метод легко можно обобщить на случай ключевого обмена большего количества участников. Использование метода Диффи - Хеллмана на практике должно сопровождаться сертификацией «открытых» ключей *X* и *Y*. Иначе злоумышленник может провести атаку, которая известна под названием «человек посередине» (man-in-the-middle), когда передаваемые участниками А и Б сообщения перехватываются злоумышленником и подменяются сообщениями *X’* и *Y’*, вычисленными на основе его закрытого ключа. В итоге будут установлены два соединения «А - зломышленник» и «Б - злоумышленник», причем А и Б будут уверены, что обмениваются сообщениями друг с другом. Необходимо также отметить, что алгоритм Диффи - Хеллмана не является асимметричным алгоритмом шифрования, шифрование при его использовании необходимо выполнять с использованием симметричного шифра. Примером действительно асимметричного алгоритма шифрования, основанного на проблеме дискретного логарифма, является алгоритм Эль-Гемаля, разработанный в 1985 г. Последовательность действий при генерации ключей, шифровании и дешифрации представлена на рис. 2.14.



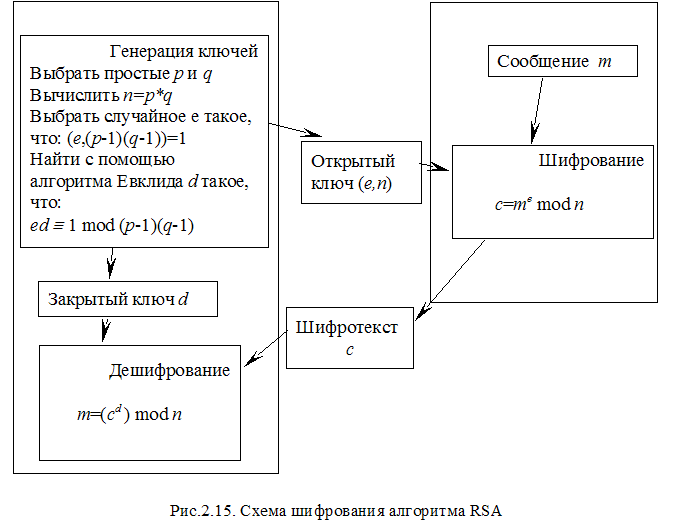
Необходимо пояснить процедуру дешифрования. Так как *ax* ≡ *gkx* mod *p*, то имеем:

*b/ax* ≡ *ykm/ax* ≡ *gxkm/gkx* = *m* mod *p*

Таким образом, кодируемое сообщение М разбивается на части, каждая из которых *m*интерпретируется как число в диапазоне [0 .. *p*-1], и выполняется операция шифрования согласно схеме на рис. 2.14. На практике при использовании данного алгоритма рекомендуется выбирать ключи размером 768, 1024 и 1536 бит.

Самым первым, действительно асимметричным алгоритмом стал алгоритм RSA, названный так по первым буквам фамилий своих разработчиков. Алгоритм был разработан в 1978 году. В основу криптостойкости RSA положена задача факторизации (разложения на множители) больших (более 200 двоичных разрядов) целых чисел.

Процедуры генерации ключей, шифрования и дешифрования для этого алгоритма представлены на рис. 2.15.



На этапе генерации ключей формируется пара ключей: закрытый *d* и открытый *e*. Шифрование данных должно начинаться с его разбиения на блоки *m* размером *k* = [log2 (*n*)] бит каждое, чтобы блок *m* можно было рассматривать как целое число в диапазоне [0 .. *n*-1]. Обратимость операции шифрования и дешифрования RSA требует доказательства. Из теоремы Эйлераизвестно, что для двух целых чисел *n* и *x*, таких, что (*n*, *x*) = 1, выполняется:

*xφ(n)* ≡ 1 mod *n*(2.2)

где φ(*n*) – функция Эйлера, значение которой равно количеству чисел меньших *n* и взаимно простых с ним. Для *n* = *p*·*q* из алгоритма RSA, где *p* и *q* – простые числа, можно записать φ(*n*) = (*p*-1)(*q*-1).

Тогда (2.2) можно переписать в виде:

*x(p-1)(q-1)* ≡ 1 mod *n*(2.3)

Возведем обе части (2.3) в степень –*y*:

*x(-y)(p-1)(q-1)* ≡ 1(-*y*) mod *n* ≡ 1 mod *n*(2.4)

Умножим обе части (2.4) на *x*:

*x(-y)(p-1)(q-1)+1* mod *n* = *x*(2.5)

Но при генерации ключей мы получили *e* и *d* такие, что *ed* ≡ 1 mod (*p*-1)(*q*-1), а это означает, что в (2.5) можно заменить 1-*y*(*p*-1)(*q*-1) на *ed*:

*xed* mod *n* = *x*

Тогда, если мы возведем шифротекста *c* = *me* mod *n* в степень *d* по модулю *n*, как мы это и делаем при дешифровании, то получим:

(*cd*) mod *n* = (*me* mod *n*)*d* mod *n* = *med* mod *n* = *m*

Очевидно, что основная задача криптоаналитика при взломе этого шифра – узнать закрытый ключ *d*. Для этого он должен выполнить те же действия, что и получатель при генерации ключа – решить в целых числах уравнение *ed* + *y*(*p*-1)(*q*-1) = 1 относительно *d* и *y*. Однако, если получателю известны входящие в уравнение параметры *p* и *q*, то криптоаналитик знает только число *n* – произведение *p* и *q*. Следовательно, ему необходимо произвести факторизацию числа *n*, то есть разложить его на множители. Для решения задачи факторизации к настоящему времени разработано множество алгоритмов: квадратичного решета, обобщенного числового решета, метод эллиптических кривых. Но для чисел большой размерностиэто очень трудоемкая задача. Ее трудоемкость можно подтвердить следующими цифрами [[11]](http://www.volpi.ru/education/zki/index.php?man=1&page=45#book11): для факторизации числа 100D (число с 100 десятичными разрядами) потребовалась вычислительная мощность 7MY (1 MY – величина, равная годовой производительности компьютера, выполняющего один миллион целочисленных инструкций в секунду), для числа 130D – 500MY, для числа 140D – 2000 MY.Современная криптография к надежным ключам шифрования RSA относит ключи длиной 768, 1024, 2048 бит.

Необходимо отметить, что математически не была доказана единственность способа восстановления *m* по *с* и *e*разложением *n* на множители. Криптоаналитики неисключают, что может быть открыт совсем иной способ криптоанализа RSA, и тогда алгоритм станет абсолютно непригодным для практического использования.

Еще одной проблемой является генерация больших простых чисел для алгоритма. Строгое доказательство простоты сгенерированного случайного числа требует решение той же самой задачи факторизации, поэтому большинство общепринятых тестов устанавливает простоту числа с некоторой вероятностью. Что произойдет, если *p* или *q* окажется составным? Тогда *у*модуля *n* будет три или более делителей. Соответственно некоторые делители будут меньше рекомендованной величины, что, в свою очередь, открывает возможности для атаки путем факторизации модуля.

Некоторые атаки используют уязвимость протокола использования алгоритма RSA [[12]](http://www.volpi.ru/education/zki/index.php?man=1&page=45#book12). Важно понимать, что само по себе использование RSA не обеспечивает требуемого уровня безопасности системы. Рассмотрим некоторые возможные атаки на протокол шифрования RSA.

Если злоумышленнику удалось перехватить сообщение *c*, зашифрованное с помощью открытого RSA-ключа пользователя А, то для раскрытия *m* = *сd* он сначала выбирает первое случайное число *r*, меньшее *n*, и затем, воспользовавшись открытым ключом А *е*, вычисляет

* *x* = *re* mod *n*,
* *y* = *xc* mod *n*,
* *t* = *r-1* mod *n*.

Если *х* = *re* mod *n*, то *r* = *xd* mod n

Далее злоумышленник каким-либо способом вынуждает А закодировать сообщение *y* на его секретном ключе. А посылает злоумышленнику

*u = yd mod n*

Теперь злоумышленник раскрывает *m*, вычисляя

*tu mod n = r-1yd mod n = r-1 xdcd mod n = cd mod n = m*

Еще одной известной атакой является атака на основе общего RSA-модуля. Если раздать всем абонентам криптосети одинаковый модуль *n*, но каждому — свои значения показателей степени (*e*1, *d*1), (*e*2, *d*2), то при шифровании одного и того же сообщения разными показателями степени (при фиксированном модуле) при условии, чтопоказатели *e*1 и *e*2 — взаимно-простые числа, открытый текст может быть раскрыт даже при неизвестных ключах дешифрования. Пусть заданы: *m* — открытый текст, *e*1 и *e*2 — два ключа шифрования, *n* — общий модуль. Шифротекстами сообщения являются:

* *c1 = me1 mod n*,
* *c2 = me2 mod n*,

Криптоаналитик знает *n*, *e*1, *e*2, *c*1 и *c*2. Так как *e*1 и *e*2 — взаимно-простые числа, то, воспользовавшись расширенным алгоритмом Евклида, можно найти такие числа *r* и *s*, что

*re1 + se2 = 1*.

Полагая *r* отрицательным (или *r*, или *s* должно быть отрицательным), можно снова воспользоваться расширенным алгоритмом Евклида для вычисления *c*1-1. Тогда

*(c1-1)-rc2s = (me1 mod n)r(me2 mod n)s = me1r+e2s mod n = m mod n*.

Таким образом, использование общего для группы пользователей параметра *n* может отрицательно сказаться на уровне безопасности криптосети.

Еще одно требование к криптопротоколам вытекает из возможности атаки «шифрование коротких сообщений». Известно, что криптосистема RSA обладает низкой криптостойкостью при зашифрованном на малом e коротком сообщении. Действительно, при *c = me < n*открытый текст *m* может быть восстановлен по шифротексту c при помощи процедуры извлечения корня. Однако меры противодействия также очевидны, — либо открытый ключ *e*должен быть достаточно большим, либо открытый текст не должен быть коротким. Выбор малого *e* обусловлен соображениями вычислительной эффективности шифрования и проверки подписи. Таким образом, разумный подход заключается в искусственном наращивании коротких открытых текстов («набивки»).

Для практической криптостойкости алгоритма RSA необходимо соблюдать еще ряд ограничений: секретный ключ *d* не должен быть слишком маленьким, числа *p* и *q* должны очень близко совпадать по порядку длины, числа (*p*+1) и (*q*+1) должны содержать в своем разложении большие простые делители. Эти ограничения учитывают ряд возможных атак на RSA, которые не рассмотрены в данном пособии.

### Достоинства и недостатки ассиметричного шифрования.

Основной недостаток симметричной криптографии -распространение ключей. Особенно, если ключ- одноразовый, как в совершенных шифрах.В идеале ключ надо менять после каждого сеанса шифрования.При этом распространение секретных ключей становится проблемой.

Асимметричная криптография эту проблему решает за счет того , что ключ фактически состоит из 2-х половинок - которые почему-то и называются в ней ключами. Одна из этих половинок называется секретным ключом, зато другая - открытый ключ.

Это означает, что из знания открытого ключа злоумышленник НЕ может извлечь МНОГО полезной информации. На самом деле частично знание открытого ключа дает некоторую информацию, НО СЛИШКОМ МАЛО, чтобы по ней можно было восстановить секретную часть.

Основных(популярных, широко известных) схем асимметричной криптографии немного - RSA , Эль-Гамаль. Каждая из них опирается на ТРУДНУЮ матемаматическую проблему.RSA - на то, что неизвестны алгоритмы практически достаточно быстрого разложения целых чисел на множители.Эль-Гамаль - на задачу нахождения так называемого дикретного логарифма.

Основные достоинства асимм. криптографии -

1) легче распространять ключи ( больше подходит для ГРАЖДАНСКОЙ криптографии )

2) теоретически их сложнее взломать ( ЕСЛИ АККУРАТНО соблюдены ВСЕ правила по работе с этими системами). Обычно для это требуется НОВЫЙ алгоритм или принципиально более быстрая реализация известного.

ЕСЛИ ПРАВИЛА НЕ СОБЛЮДЕНЫ - то к сожалению у злоумышленника ПОЯВЛЯЕТСЯ шанс.

Недостатки ассиметричного шифрования:.

1) реализации асимм. криптографии - существенно - на 2-4 порядка медленее, чем симм.

2) алгоритмы понимаются с большим трудом ( надо знать дискр. математику - операции по модулю, поля Галуа, и т.п.)

3) это молодая отрасль криптографии.

4) алгоритмы требуется АККУРАТНО реализовывать ( см. Шнаера.)

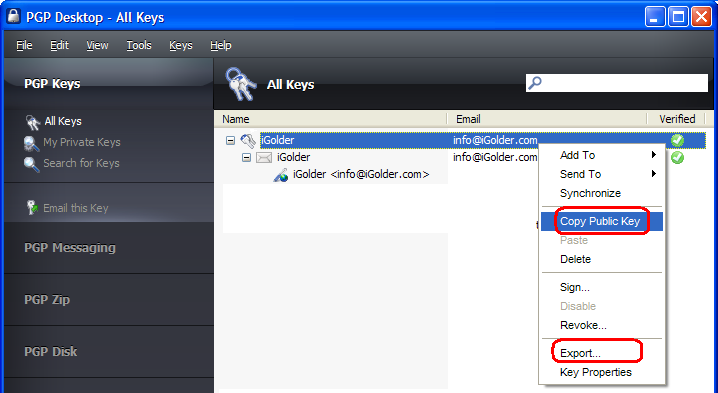
Ввиду всего вышеперечисленного обычно асимм. криптографию используют ВМЕСТЕ с симм. крипт. - фактически по гибридной схеме. Например как в PGP - асимм. шифрует ключи для симм. криптографии. тем самым пытаются совместить плюсы обоих схем и хоть частично нивелировать минусы.

# Практические задания

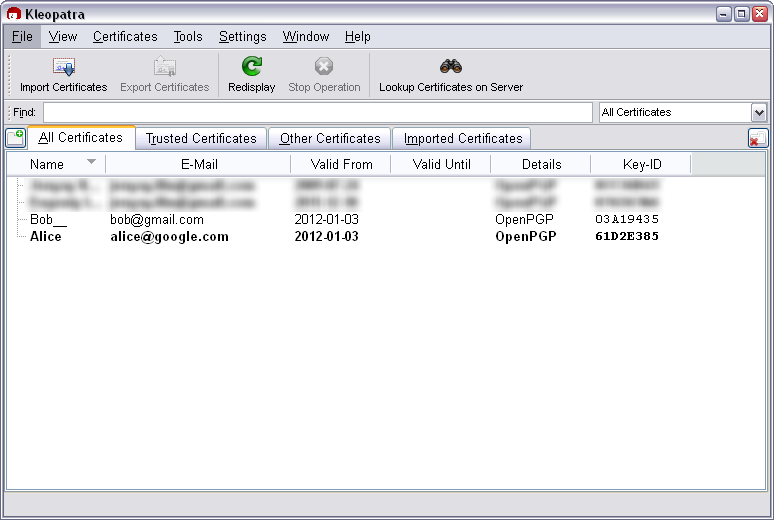
1. Установить (или запрограммировать) программу, реализующую ассиметричное шифрование.
2. Практически осуществить кодирование и раскодирование.
3. Сделать выводы. Оформить лабораторную работу.

### Пример выполнения практической части

Предполагается использование PGP или другой программы на её основе.



Использование PGP Desktop



Использование Kleopatra

# Контрольные вопросы

1. Определение асимметричного шифрования и его отличия от симметричного

2. Схема шифрования с открытым ключом

3. Преимущества и недостатки по отношению к симметричным

4. Как работает алгоритм RSA.

**Литература.**

1. Скляров. Хакерские фишки.

2. Xaker.ru

3. Алферов, Зубов, Кузьмин, Черемушкин. Основы криптографии.2-е изд.М.Гелиос.2002 г.

4. Википедия.

5. Д. Н. Лясин, С. Г. Саньков Методы и средства защиты компьютерной информации.