БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

МЕХАНИКО-МАТЕМАТИЧЕСКИЙ ФАКУЛЬТЕТ

Кафедра дифференциальных уравнений и системного анализа

КАТЛИНСКИЙ Илья Геннадьевич

КРИПТОСИСТЕМА RSA

Дипломная работа

Студента V курса специализации 1-31 03 01-06 01 -

на соискание квалификации “Математик.Системный аналитик.”

Руководитель

ЧЕРГИНЕЦ Дмитрий Николаевич

доцент кафедры ДУиСА

Допустить к защите

Заведующий кафедрой,

профессор

Минск, 2013

Оглавление

Глава 1

Введение

1.1 Основные определения

Криптография - область знаний, которая занимается разработкой методов преобразования информации с целью обеспечения ее конфиденциальности, целостности и аутентификации.

Пусть A и B - конечные множества, будем называть их алфавитами. Информацию, состоящую из конечного объединения элементов множества A, которую будем защищать, будем называть *открытым текстом*. Конечное объединение элементов множества B будем называть *шифротекстом*. Пусть X и Y - множества открытых текстов и шифрованых текстов соответственно.

Функцию $E_k : X \to Y$, где k - параметр функции,который будем называть ключом, принадледит множеству ключей K, будем называть *функцией шифрования*.

Функция $D_k : Y \to X$ называется *функцией дешифрования*. *Шифром* или *криптосистемой* называется набор $(A, B, X, Y, K, E_k, D_k)$, удовлетворяющий требованию $D_k(E_k(x)) = x$ для каждого $x \subseteq X$ и $k \subseteq K$ *Шифрование* - процесс применения шифра к защищаемой информации, преобраование информации *(открытого текста)* в шифрованное сообщение *(шифротекст)* с помощью определенных правил, содержщихся в шифре. *Дешифрование* - процесс, обратный *шифрованию*, преоразрвание шифрованного сообщения в защищаемую информацию с помощью определенных правил, содержащихся в шифре. Криптосистемы (X, Y, K, E_k, D_k), в которых в функции шифрования E_k и в функции дешифрования D_k используется один и тот же ключ $k \subseteq K$, называется симметричным. Шифры,в которых для штфрования используется один ключ, а для расщифрования - другой, называются ассиметричными или криптосистемами с открытым ключом. Таким образом, криптосистемой с открытым ключом называется система $(X, Y, (k_e, k_d) \subseteq K, E_{k_e}, D_{k_e,k_d})$, где алгорит-

мы шифрования и дешифрования являются открытыми, шифрованный текст C и открытый ключ k_e могут передаваться по незащищенному каналу, секретный ключ k_d является секретным.

Основные требования, которые предъялвяются к криптосистемам с открытым ключом:

1. Вычисление пары $(k_e,\ k_d)$ получателем должно быть простым (полиномиальный алгоритм).

2. Отправитель, знаю открытый ключ k_e и сообщение m, может легко вычислить криптограмму $c = E_{k_e}(m)$.

3. Получатель, используя секретный ключ k_d и криптограмму c, может легко восстановить исходное сообщение $m = D_{k_d}(c)$.

4. Противник, зная открытый ключ k_e, при попытке вычислить секретный ключ k_d не может его высилить

5. Противник, зная пару $(k_e,\ c)$, при попытке вычилить исходное сообщение m не может его вычислить

1.2 Введение в теорию шифрования

Труды Евклида и Диофанта, Ферма и Эйлера, Гаусса, Чебышева и Эрмита содержат остроумные и весьма эффективные алгоритмы решения диофантовых уравнений, выяснения разрешимости сравнений, построения больших по тем временам простых чисел, нахождения наилучших приближений и т.д. В последние два десятилетия, благодаря в первую очередь запросам криптографии и широкому распространению ЭВМ, исследования по алгоритмическим вопросам теории чисел переживают период бурного и весьма плодотворного развития. Вычислительные машины и электронные средства связи проникли практически во все сферы человеческой деятельности. Немыслима без них и современная криптография. Шифрование и дешифрование текстов можно представлять себе как процессы переработки целых чисел при помощи ЭВМ, а способы, которыми выполняются эти операции, как некоторые функции, определённые на множестве целых чисел. Всё это делает естественным

появление в криптографии методов теории чисел. Кроме того, стойкость ряда современных криптосистем обосновывается только сложностью некоторых теоретико-числовых задач. Но возможности ЭВМ имеют определённые границы. Приходится разбивать длинную цифровую последовательность на блоки ограниченной длины и шифровать каждый такой блок отдельно. Мы будем считать в дальнейшем, что все шифруемые целые числа неотрицательны и по величине меньше некоторого заданного (скажем, техническими ограничениями) числа m. Таким же условиям будут удовлетворять и числа, получаемые в процессе шифрования. Это позволяет считать и те, и другие числа элементами кольца вычетов. Шифрующая функция при этом может рассматриваться как взаимнооднозначное отображение колец вычетов а число представляет собой сообщение в зашифрованном виде.

Простейший шифр такого рода - шифр замены, соответствует отображению

$$f : x \to x + k \ (mod\ m) \tag{1.1}$$

при некотором фиксированном целом k. Подобный шифр использовал еще Юлий Цезарь. Конечно, не каждое отображение подходит для целей надежного сокрытия информации.

В 1978 г. американцы Р. Ривест, А. Шамир и Л. Адлеман (R.L.Rivest. A.Shamir. L.Adleman) предложили пример функции f, обладающей рядом замечательных достоинств. На её основе была построена реально используемая система шифрования, получившая название по первым буквам имен авторов - система RSA. Эта функция такова, что

1. существует достаточно быстрый алгоритм вычисления значений $f(x)$;

2. существует достаточно быстрый алгоритм вычисления значений обратной функции $f^{-1}(x)$;

3. функция $f(x)$ обладает некоторым «секретом», знание которого позволяет быстро вычислять значения $f^{-1}(x)$; в противном же случае вычисление $f^{-1}(x)$ становится трудно разрешимой в вычислительном отношении задачей, требующей для своего решения столь много времени, что по его прошествии зашифрованная информация перестает представлять интерес для лиц, использующих отображение f в качестве шифра.

Еще до выхода из печати статьи копия доклада в Массачусетском Технологическом институте, посвящённого системе RSA. была послана известному популяризатору математики М. Гарднеру, который в 1977 г. в журнале Scientific American опубликовал статью посвящённую этой системе шифрования. В русском переводе заглавие статьи Гарднера звучит так: Новый вид шифра, на расшифровку которого потребуются миллионы лет. Именно эта статья сыграла важнейшую роль в распространении информации об RSA, привлекла к криптографии внимание широких кругов неспециалистов и фактически способствовала бурному прогрессу этой области, произошедшему в последовавшие 20 лет.

1.3 Постановка задачи

Глава 2

Простые числа

2.1 Построение больших простых чисел

Существует довольно эффективный способ убедиться, что заданное число является составным, не разлагая это число на множители. Согласно малой теореме Ферма, если число n простое, то для любого целого a, не делящегося на n, выполняется сравнение

$$a^{n-1} \equiv 1 \ (mod\ n). \tag{2.1}$$

Если же при каком-то a это сравнение нарушается, можно утверждать, что n - составное. Вопрос только в том, как найти для составного n целое число a, не удовлетворяющее (***). Можно, например, пытаться найти необходимое число a, испытывая все целые числа подряд, начиная с 2. Или попробовать выбирать эти числа случайным образом на отрезке $1 < a < n$.

К сожалению, такой подход не всегда даёт то, что хотелось бы. Имеются составные числа n, обладающие свойством (***) для любого целого a с условием $(a,\ n) = 1$. Такие числа называются числами Кармайкла. Рассмотрим, например, число $561 = 3 \cdot 11 \cdot 17$. Так как 560 делится на каждое из чисел 2, 10, 16, то с помощью малой теоремы Ферма легко проверить, что 561 есть число Кармайкла. Можно доказать, что любое из чисел Кармайкла имеет вид $n = p_1 \cdot \ldots \cdot p_r,\ r \geq 3$, где все простые p_i различны, причем $n - 1$ делится на каждую разность $p_i - 1$. Лишь недавно, была решена проблема о бесконечности множества таких чисел.

В 1976 г. Миллер предложил заменить проверку (***) проверкой несколько иного условия. Если n - простое число, $n - 1 = 2^s t$, где t нечётно, то согласно малой теореме Ферма для каждого a с условием $(a,\ n) = 1$ хотя бы одна из скобок в произведении

$$(a^t - 1)(a^t + 1)(a^{2t} + 1) \cdot \ldots \cdot (a^{2^{s-1}t} + 1) = a^{n-1} - 1 \tag{2.2}$$

делится на n. Обращение этого свойства можно использовать, чтобы отличать составные числа от простых.

Пусть n - нечётное составное число, $n - 1 = 2_s\ t$, где t нечётно. Назовем целое число a, $1 < a < n$, «хорошим» для n, если нарушается одно из двух условий:

1. n не делится на a;
2. $a^{\,t} \equiv 1\ (mod\ n)$ или существует целое k, $0 \leq \mathrm{k} < \mathrm{s}$, такое, что

$$a^{\,2^{\,k}\,t} \equiv -1\ (mod\ n) \tag{2.3}$$

Из сказанного ранее следует, что для простого числа n не существует хороших чисел a. Если же n составное число, то, как доказал Рабин, их существует не менее $\frac{3}{4}(n - 1)$.

2.1.1 Проверка большого числа на простоту

Есть некоторое отличие в постановках задач предыдущего и настоящего пунктов. Когда мы строим простое число n, мы обладаем некоторой дополнительной информацией о нем, возникающей в процессе построения. Например, такой информацией является знание простых делителей числа $n - 1$. Эта информация иногда облегчает доказательство простоты n.

В настоящее время известны детерминированные алгоритмы различной сложности для доказательства простоты чисел. К примеру алгоритм Адлемана, Померанца и Рамели. Для доказательства простоты или непростоты числа n этот алгоритм требует $(\ln n)^{\,C \ln \ln \ln n}$ арифметических операций. Здесь C - некоторая положительная абсолютная постоянная. Функция $\ln \ln \ln n$ хоть и медленно, но всё же возрастает с ростом n, поэтому алгоритм не является полиномиальным. Но всё же его практические реализации позволяют достаточно быстро тестировать числа на простоту.

2.1.2 Примеры

2.2 Проблема факторизации

2.2.1 Метод Полларда

2.2.2 Метод квадратов

2.2.3 Обобщенный метод Ферма

2.2.4 Метод Диксона

2.2.5 Метод квадратичного решета

2.2.6 Метод решета числового поля

Глава 3

RSA

3.1 Основы RSA

Пусть n и e натуральные числа. Функция f реализующая схему RSA, устроена следующим образом

$$f : x \to x^e \ (mod\ n), \tag{3.1}$$

Для расшифровки сообщения $a = f(x)$ достаточно решить сравнение

$$x^e = a \ (mod\ n) \tag{3.2}$$

При некоторых условиях на n и e это сравнение имеет единственное решение x.

Для того, чтобы описать эти условия и объяснить, как можно найти решение, нам потребуется одна теоретико-числовая функция - функция Эйлера. Эта функция натурального аргумента n обозначается $\varphi(n)$ и равняется количеству целых чисел на отрезке от 1 до n, взаимно простых с n. Так $\varphi(1) = 1$ и $\varphi(p^r) = p^{r-1}(p-1)$ для любого простого числа p и натурального r. Кроме того, $\varphi(a\ b) = \varphi(b)\ \varphi(a)$ для любых натуральных взаимно простых a и b. Эти свойства позволяют легко вычислить значение $\varphi(n)$, если известно разложение числа n на простые сомножители.

Если показатель степени e в сравнении (3.2) взаимно прост с $\varphi(n)$, то сравнение (3.2) имеет единственное решение. Для того, чтобы найти его, определим целое число d, удовлетворяющее условиям.

$$d\ e \equiv (mod\ \varphi(n)),\ 1 \leq d < \varphi. \tag{3.3}$$

Такое число существует, поскольку $(e,\ \varphi(n)) = 1$, и притом единственно. Здесь и далее символом $(a,\ b)$ будет обозначаться наибольший общий делитель чисел a и b. Классическая теорема Эйлера, утверждает, что для каждого числа x, взаимно

простого с n, выполняется сравнение $x^{\varphi(n)} \equiv 1 \ (mod\ n)$ и, следовательно

$$a^{d} \equiv x^{d\,e} \equiv x \ (mod\ n). \tag{3.4}$$

Таким образом, в предположении $(a,\ m) = 1$, единственное решение сравнения (3.2) может быть найдено в виде

$$x \equiv a^{d} \ (mod\ n). \tag{3.5}$$

Если дополнительно предположить, что число n состоит из различных простых сомножителей, то сравнение (3.5) будет выполняться и без предположения $(a,\ m) = 1$. Действительно, обозначим $r = (a,\ n)$ и $s = n/r$. Тогда φ(n) делится на φ(r), а из (3.2) следует, что $(x,\ s) = 1$. Подобно (3.4), теперь легко находим (3.5). А кроме того, имеем $x \equiv 0 \equiv a^{r} \ (mod\ r)$. Получившиеся сравнения в силу $(r,\ s) = 1$ дают нам (3.5).

Функция (3.1), принятая в системе RSA, может быть вычислена достаточно быстро. Обратная к $f(x)$ функция $f^{-1} : x \to x^{d} \ (mod\ n)$ вычисляется по тем же правилам, что и $f(x)$, лишь с заменой показателя степени e на d.

Для вычисления функции (3.1) достаточно знать лишь числа e и n. Именно они составляют открытый ключ для шифрования. А вот для вычисления обратной функции требуется знать число d. Казалось бы, ничего не стоит, зная число n, разложить его на простые сомножители, вычислить затем с помощью известных правил значение $\varphi(n)$ и, наконец, с помощью (3.3) определить нужное число d. Все шаги этого вычисления могут быть реализованы достаточно быстро, за исключением первого. Именно разложение числа n на простые множители и составляет наиболее трудоемкую часть вычислений. В теории чисел несмотря на многолетнюю её историю и на очень интенсивные поиски в течение последних 20 лет, эффективный алгоритм разложения натуральных чисел на множители так и не найден.

Авторы схемы RSA предложили выбирать число n в виде произведения двух простых множителей p и q, примерно одинаковых по величине. Так как

$$\varphi(n) = \varphi(p\ q) = (p\text{-}1)(q\text{-}1), \tag{3.6}$$

то единственное условие на выбор показателя степени e в отображении (1) есть

$$(e,\ p\text{-}1) = (e,\ q\text{-}1) = 1 \tag{3.7}$$

Итак, лицо, заинтересованное в организации шифрованной переписки с помощью схемы RSA, выбирает два достаточно больших простых числа p и q. Перемножая их, оно находит число $n = p\ q$. Затем выбирается число e, удовлетворяющее условиям (3.7), вычисляется с помощью (3.6) число $\varphi(n)$ и с помощью (3.3) - число d. Числа n и e публикуются, число d остается секретным.

Для иллюстрации своего метода Ривест, Шамир и Адлеман зашифровали таким способом некоторую английскую фразу. Сначала она стандартным образом (a=01, b=02, ... z=26, пробел=00) была записана в виде целого числа x, а затем зашифрована с помощью отображения (3.1) при

$$m=114381625757888867669325779976146612010218296721242362562561842935706935245733897830597123563958705058989075147599290026879543541$$

и

$$e=9007.$$

Эти два числа были опубликованы, причем дополнительно сообщалось, что $n = p\ q$. где p и q - простые числа, записываемые соответственно 64 и 65 десятичными знаками.

Сложность алгоритмов теории чисел обычно принято измерять количеством арифметических операций (сложений, вычитаний, умножений и делений с остатком), необходимых для выполнения всех действий, предписанных алгоритмом. Впрочем, это определение не учитывает величины чисел, участвующих в вычислениях. Ясно, что перемножить два стозначных числа значительно сложнее, чем два однозначных, хотя при этом и в том, и в другом случае выполняется лишь одна арифметическая операция. Поэтому иногда учитывают ещё и величину чисел, сводя дело к так называемым битовым операциям, т. е. оценивая количество необходимых операций с цифрами 0 и 1, в двоичной записи чисел. Говоря о сложности алгоритмов, мы будем иметь в виду количество арифметических операций. При построении эффективных алгоритмов и обсуждении верхних оценок сложности обычно хватает интуитивных понятий той области математики, которой принадлежит алгоритм. Формализация же этих понятий требуется лишь тогда, когда речь идёт об отсутствии алгоритма или доказательстве нижних опенок сложности. Приведем теперь примеры достаточно быстрых алгоритмов с опенками их сложности. Здесь и в дальнейшем мы не

будем придерживаться формального описания алгоритмов, стараясь в первую очередь объяснить смысл выполняемых действий.

3.1.1 Алгоритм создания ключей

3.1.2 Алгоритм шифрования и расшифрования

3.1.3 Примеры

3.2 Криптоанализ RSA

3.2.1 LLL-алгоритм

3.2.2 Теорема Копперсмита

3.3 Атаки на RSA

3.3.1 Relaxed RSA problem

3.3.2 Атака Франклина-Райтера

3.3.3 Расширенная атака Хастаадта

3.3.4 Factoring with High Bits Known

3.3.5 Атака Винера

3.3.6 Циклическая атака

3.4 Применение RSA

Крайне не рекомендуется использовать *RSA* в его наиболее простой форме, существуют требования, выдвигаемые к асимметричным криптосистемам.

3.4.1 Пример 1

Современная асимметричная криптосистема может считаться стойкой, если злоумышленник, имея два открытых текста M_1 и M_2, а также один шифротекст C_b не может с вероятностью большей, чем 0.5 определить какому из двух открытых текстов соответствует шифротекст C_b.

Проверим, удовлетворяет ли RSA данному требованию. Пусть злоумышленник прослушивает переписку A и B. Злоумышленник видит, что B в открытом виде задал A вопрос. A односложно отвечает B на этот вопрос. A шифрует свой ответ открытым ключом B и отправляет шифротекст. Далее злоумышленник перехватывает шифротекст и подозревает, что в нем зашифровано либо Да, либо Нет. Всё, что ему теперь нужно сделать для того чтобы узнать ответ A это зашифровать открытым ключом B слово Да и если полученный криптотекст совпадет с перехваченным, то это означает, что A ответила Да, в противном же случает злоумышленник поймет, что ответом было Нет.

Как видно из примера, RSA не столь надежна как это принято считать. Чтобы избежать подобных ситуаций, достаточно чтобы алгоритм добавлял к тексту некоторую случайную информацию, которую бы невозможно было предугадать.

3.4.2 Пример 2

Рассмотрим следующий пример: пусть злоумышленник имеет доступ к расшифровывающему «черному ящику». Таким образом любой криптотекст по просьбе злоумышленника может быть расшифрован. Далее злоумышленник создает два открытых текста M_1 и M_2. Один из этих текстов шифруется и полученный в результате криптотекст C_b возвращается злоумышленнику. Задача злоумышленника угадать с вероятностью большей чем 0.5 какому из сообщений M_1 и M_2 соответсвует криптотекст C_b. При этом он может попросить расшифровать любое сообщение, кроме C_b. Говорят что криптосистема стойкая, если злоумышленник, даже в таких прекрасных для себя условиях, не сможет указать какому исходному тексту соответствует C_b с вероятностью большей 0.5.

Рассмотрим насколько криптостойкой окажется RSA в данном случае. Итак, злоумышленник имеет два сообщения M_1 и M_2. А также криптотекст $C_b = M_1^e \pmod n$.

Ему необходимо указать какому конкретно из двух текстов соответствует C_b. Для этого он может предпринять следующее. Зная открытый ключ e, он может создать сообщение $C'=2^e\ C_b(mod\ n)$. Далее он просит расшифровывающий «черный ящик» расшифровать сообщение C'. А затем несложная арифметика ему в помощь. Имеем:

$$M'=C'^d(mod\ n)=2^{e\ d}M_1^{e\ d}(mod\ n)=2M_1(mod\ n).$$

Таким образом вычислив $M'/2$ злоумышленник увидит M_1. А это означает, что он поймет что в нашем примере было зашифровано сообщение M_1, а следовательно мы еще раз убедились в неприемлемости использования *RSA* в его изначальном виде на практике.

3.4.3 RSA-OAEP

Таким образом, уже сейчас можно сказать, что *RSA* во всех своих проявлениях будь то *PGP* или *SSL* не шифрует только отправленные на вход шифрующей функции данные. Алгоритм сперва добавляет к этим данным блоки содержащие случайный набор бит. И только после этого полученный результат шифруется. Это значит, что вместо привычной всем $c=m^e(mod\ n)$ получаем более близкую к действительности $c=(m//r)^e(mod\ n)$, где r - случайное число. Такую методику называют схемами дополнения. В настоящее время использование *RSA* без схем дополнения является не столько плохим тоном, сколько непосредственно нарушением стандартов.

Устранить и эту неприятность помогают схемы дополнения. Только теперь к ним выдвигается требование не только о том, чтобы дополнительная информация была абсолютно случайной и непрогнозируемой. Но так же и том, чтобы дополнительные блоки помогали определить был ли шифротекст получен в результате работы шифрующей функции или он смоделирован злоумышленником. Причем в случае, если будет обнаружено, что шифротекст смоделирован вместо расшифрованных данных атакующему будет выдано сообщение о несоответствие данных реальному криптотексту.

В *RSA* при подписи и при шифровании данных используют две различные схемы дополнений. Схема, реализуемая для подписания документов, называется *RSA-PSS(probabilistic signature scheme)* или вероятностная схема подписи. Схема, используемая при шифровании – *RSA-OAEP(Optimal asymmetric encryption padding)* или

оптимизированное асимметричное дополнение шифрования, на примере *OAEP* и рассмотрим как на самом деле происходит шифрование сообщений в *RSA*.

Итак чтобы зашифровать абсолютно любое сообщение в *RSA-OAEP* делается следующее:
Выбираются две хеш-функции *G(x)* и *H(x)* таким образом, чтобы суммарная длина результатов хеш-функций не превышала длины ключа *RSA*. Генерируется случайная строка битов *l*. Длина строки должна быть равна длине результата хеш-функции *H(x)*.

Итак чтобы зашифровать абсолютно любое сообщение в *RSA-OAEP* делается следующее:

1. Выбираются две хеш-функции *G(x)* и *H(x)* таким образом, чтобы суммарная длина результатов хеш-функций не превышала длины ключа *RSA*.
2. Генерируется случайная строка битов *l*. Длина строки должна быть равна длине результата хеш-функции *H(x)*.
3. Сообщение *M* разбивают на блоки по *k-бит*. Затем к каждому полученному блоку *m* дописывают *(n-k)* нулей. Где *n-длина* хеш-функции *G(x)*.
4. Определяют следующий набор бит: $\{\mathrm{m}||0^{(\mathrm{n-k})} \oplus \mathrm{G(l)}\}||\{\mathrm{l} \oplus \mathrm{H(m}||0^{(\mathrm{n-k})} \oplus \mathrm{G(l))}\}$
5. Полученные биты представляют в виде целого числа M_1
6. Криптотекст получают по формуле: $C=M_1^e (mod\ n)$

Процесс дешифрования выглядит следующим образом:

1. Находят M_1 по формуле $M_1=C^d (mod\ n)$
2. В полученном наборе бит отсекают левую часть. В смысле: левой частью служат n левых бит числа M_1 где n-длина хеш-функции *G(x)*. Обозначим эти биты условно *T*. И заметим, что $T=\{m||0^{(n-k)} \oplus G(l)\}$. Все остальные биты являются правой частью.
3. Находим $H(T)=H(m||0^{(n-k)}\ G(l))$
4. Зная H(T) получаем l, поскольку знаем $\mathrm{l} \oplus \mathrm{H(T)}$-это правая часть блока

5. Вычислив l, находим m из T$\oplus$G(l), поскольку T=$\{m||0^{(n-k)}\oplus G(l)\}$

6. Если *m* заканчивается *(n-k)*-нулями значит сообщение зашифровано правильно. Если нет то это значит, что шифротекст некорректен, а следовательно он скорее всего подделан злоумышленником.

Таким образом *RSA* это не только возведение в степень по модулю большого числа. Это еще и добавление избыточных данных позволяющих реализовать дополнительную защиту вашей информации. Вы, возможно, спросите: а зачем это все нужно? Неужели в действительности может произойти такая ситуация, когда атакующий получит доступ к расшифровывающему алгоритму? Совсем по другому поводу как-то было сказано: если какая-либо неприятность может произойти, она обязательно произойдет.

3.5 Обобщение RSA

Глава 4

Заключение

4.1 Выводы

4.2 Список литературы