Лабораторная работа № 5

Обмен ключами по схеме Диффи-Хеллмана

Цель работы

Освоить методы генерации больших простых чисел и методы проверки больших чисел на простоту. Познакомиться с теоремой Эйлера, научиться строить первообразные корни по модулю n. Изучить схему обмена ключами Диффи-Хеллмана.

Генерация большого простого числа

Любая криптосистема основана на использовании ключей. Если для обеспечения конфиденциального обмена информацией между двумя пользователями процесс обмена ключами тривиален, то в системе, в которой количество пользователей составляет десятки и сотни, управление ключами – серьёзная проблема. Если не обеспечено достаточно надёжное управление ключевой информацией, то, завладев ею, злоумышленник получает неограниченный доступ ко всей информации. В этом случае необходимо введение какой-либо случайной величины в процесс шифрования.

В частности, для реализации алгоритма RSA требуются большие простые числа. Где их взять?

Простых чисел не так мало, как кажется. Например, существует приблизительно 10^{151} простых чисел длиною от 1 бита до 512 включительно. Для чисел, близких к n, вероятность того, что выбранное число окажется простым, равна $\frac{1}{\ln n}$. Поэтому полное число простых чисел, меньших n, равно $\frac{n}{\ln n}$. Считается, что вероятность выбора двумя людьми одного и того же большого простого числа пренебрежимо мала.

Существуют различные вероятностные проверки чисел на простоту, определяющие с заданной степенью достоверности, является ли число простым. При условии, что эта степень достоверности велика, такие способы достаточно хороши. Такие простые числа часто называют «промышленными простыми», т.е. они просты с контролируемой возможностью ошибки.

Повсеместно используется алгоритм, разработанный Майклом Рабином по идеям Гари Миллера.

Тест Рабина-Миллера

Выбрать для проверки случайное число p [1, 2]. Вычислить b как наибольшее число делений $p-1$ на 2, т.е. b – наибольший показатель степени числа 2, на которую делится $p-1$ без остатка. Затем вычислить m такое, что $p = 1 + 2^b \cdot m$.

1. Выбрать случайное число a, меньшее p.
2. Установить $j = 0$ и $z = a^m \bmod p$.
3. Если $z = 1$ или $z = p - 1$, то p проходит проверку и может быть простым числом.
4. Если $j > 0$ и $z = 1$, то p не является простым числом.
5. Установить $j = j + 1$.
   Если $j < b$ и $z < p - 1$, установить $z = z^2 \bmod p$ и вернуться на пункт 4.
   Если $z = p - 1$, то p проходит проверку и может быть простым числом.
6. Если $j = b$ и $z \neq p - 1$, то p не является простым числом.

Повторить эту проверку нужно t раз.

Доказано, что в этом тесте вероятность прохождения проверки составным числом убывает быстрее, чем в прочих. Гарантируется, что 75 % возможных значений a окажутся показателями того, что выбранное число p – составное. Это значит, что вероятность принять составное число p за простое не превышает величины $\left(\frac{1}{4}\right)^t$.

Алгоритм генерации простого числа

1. Сгенерировать случайное n-битное число p.
2. Установить его старший и младший биты равными 1. Старший бит будет гарантировать требуемую длину искомого числа, а младший бит – обеспечивать его нечётность.
3. Убедиться, что p не делится на небольшие простые числа: 3, 5, 7, 11 и т.д. Наиболее эффективна проверка на делимость на все простые числа, меньшие 2000.
4. Выполнить тест Рабина-Миллера минимум 5 раз.

Если p не прошло хотя бы одну проверку в пунктах 3 или 4, то оно не является простым.

Проверка, что случайное нечётное p не делится на 3, 5 и 7, отсекает 54 % нечётных чисел. Проверка делимости на все простые числа, меньшие 256, отсекает 80 % составных нечётных чисел.

Даже если составное число «просочилось» через этот алгоритм, это будет сразу же замечено, т.к. шифрование и дешифрование не будут работать [1, 2].

Построение первообразного корня по модулю n

В силу теоремы Эйлера для любых взаимно простых a и n выполняется соотношение:

$$a^{\varphi(n)} \equiv 1 \pmod{n}, \tag{1}$$

где $\varphi(n)$ обозначает ***функцию Эйлера***, значение которой равно количеству положительных целых значений, меньших n и взаимно простых с n. Для простого числа p выполняется:

$$\varphi(p) = p - 1.$$

Если предположить, что два числа p и q – простые, тогда для $n = p^{\alpha} \cdot q^{\beta}$ функция Эйлера будет иметь вид:

$$\varphi(n) = \varphi\left(p^{\alpha} \cdot q^{\beta}\right) = \varphi(p^{\alpha}) \cdot \varphi\left(q^{\beta}\right) = [(p-1) \cdot p^{\alpha-1}] \cdot \left[(q-1) \cdot q^{\beta-1}\right]. \tag{2}$$

Рассмотрим более общее соотношение, чем (1).

Говорят, что число a, взаимно простое с модулем n, ***принадлежит показателю*** m, если m – такое наименьшее натуральное число, что выполняется сравнение:

$$a^{m} \equiv 1 \pmod{n}. \tag{3}$$

Если a и n – взаимно простые, то существует, по крайней мере, одно число $m = \varphi(n)$, удовлетворяющее (3). Наименьшее из положительных чисел m, для которых выполняется (3), является длиной периода последовательности, генерируемой степенями a.

Справедливы следующие свойства [1].

Свойство 1. Числа $a^0, a^1, \dots, a^{m-1}$ попарно несравнимы по модулю n.

Свойство 2. $a^{\gamma} \equiv a^{\gamma'} \pmod{n} \Leftrightarrow \gamma \equiv \gamma' \pmod{m}$. *Доказательство*: разделим γ и γ' на m с остатками: $\gamma = m \cdot q + r,\ \gamma' = m \cdot q' + r'$. Тогда $a^{\gamma} \equiv a^{\gamma'} \Leftrightarrow a^{m \cdot q + r} \equiv a^{m \cdot q' + r'} \Leftrightarrow a^{r} \equiv a^{r'} \Leftrightarrow r = r'$.

Отсюда вытекает следующее свойство.

Свойство 3. Число a, принадлежащее показателю $\varphi(n)$, называется ***первообразным корнем по модулю n***.

Свойство 4. По любому простому модулю p существует первообразный корень. Первообразные корни существуют по модулям 2, 4, p^{α}, $2 \cdot p^{\alpha}$, где p – нечётное простое число, а $\alpha \in N$.

Свойство 5. Пусть $c = \varphi(n)$ и $q_1, q_2, \dots, q_k$ – различные простые делители числа c. Число a, взаимно простое с модулем n, будет первообразным корнем тогда и только тогда, когда не выполнено ни одно из следующих сравнений:

$$a^{c/q_1} \equiv 1 \pmod{n},\ a^{c/q_2} \equiv 1 \pmod{n}, \dots, a^{c/q_k} \equiv 1 \pmod{n}.$$

Доказательство. Необходимость следует из того, что $a^{\varphi(n)} \equiv 1 \pmod{n}$ и сравнение не имеет места при меньших показателях степени.

Достаточность: допустим, что a не удовлетворяет ни одному из сравнений и пусть a принадлежит показателю $m < c$. Тогда $m \mid c \Rightarrow c = m \cdot u$. Обозначим через q простой делитель u. Тогда легко получить противоречие:

$$a^{c/q} = a^{mu/q} = (a^m)^{u/q} \equiv 1 \pmod{n}.$$

Если некоторая последовательность имеет длину $\varphi(n)$, тогда целое число a генерирует своими степенями множество всех ненулевых вычетов по модулю n. Такое целое число называют ***первообразным корнем по модулю n***. Для числа n их количество равно $\varphi(\varphi(n-1))$, где $\varphi()$ – функция Эйлера.

Пример. Пусть $n = 41$. Имеем $c = \varphi(41) = 40 = 2^3 \cdot 5$. Итак, первообразный корень не должен удовлетворять двум сравнениям:

$$a^8 \equiv 1 \pmod{41}, \quad a^{20} \equiv 1 \pmod{41}.$$

Испытываем числа 2, 3, 4, … : $2^8 \equiv 10$, $2^{20} \equiv 1$, $3^8 \equiv 1$, $4^8 \equiv 18$, $4^{20} \equiv 1$, $5^8 \equiv 18$, $5^{20} \equiv 1$, $6^8 \equiv 10$, $6^{20} \equiv 40$. Отсюда видно, что 6 – наименьший первообразный корень по модулю 41.

Алгоритм обмена ключами по схеме Диффи-Хеллмана

В 1976 году была опубликована работа молодых американских математиков У. Диффи и М.Э. Хеллмана «Новые направления в криптографии». В ней они предложили конкретную конструкцию так называемого «открытого распределения ключей» [2, 3].

Цель алгоритма состоит в том, чтобы два участника могли безопасно обменяться ключом, который в дальнейшем может использоваться в каком-либо алгоритме симметричного шифрования. Сам алгоритм Диффи-Хеллмана может применяться только для обмена ключами. Алгоритм основан на трудности вычислений дискретных логарифмов.

Безопасность обмена ключами в алгоритме Диффи-Хеллмана вытекает из того факта, что, хотя относительно легко вычислить экспоненты по модулю простого числа, но очень трудно вычислить дискретные логарифмы. Для больших простых чисел задача считается неразрешимой.

Предположим, что двум абонентам необходимо провести конфиденциальную переписку, а в их распоряжении нет первоначально оговоренного секретного ключа. Однако между ними существует канал, защищённый от модификации, т.е. данные, передаваемые по нему, могут быть прослушаны, но не изменены (такие условия имеют место довольно часто). В этом случае две стороны могут создать одинаковый секретный ключ, ни разу не передав его по сети, по следующему алгоритму (см. рисунок 1).

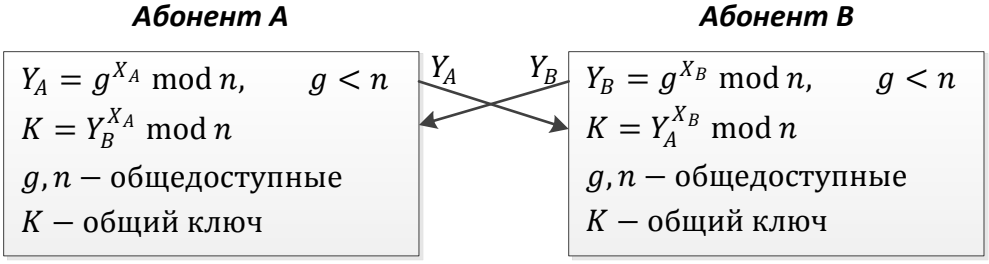

Рисунок 1 – Обмен ключами по схеме Диффи-Хеллмана

Алгоритм заключается в следующем:

1. Задаются глобальные открытые элементы:
   1) n – случайное большое простое число;
   2) g – первообразный корень n.
2. Вычисляется ключ абонентом A:
   1) выбирается большое секретное число X_A ($X_A < n$);
   2) вычисление открытого значения Y_A: $Y_A = g^{X_A} \bmod n$.
3. Вычисляется ключ абонентом B:
   1) выбирается большое секретное число X_B ($X_B < n$);
   2) вычисление открытого значения Y_B: $Y_B = g^{X_B} \bmod n$.

4. Вычисляется секретный ключ абонентом A: $K = Y_B^{X_A} \bmod n$.
5. Вычисляется секретный ключ абонентом B: $K = Y_A^{X_B} \bmod n$.

Необходимо ещё раз отметить, что алгоритм Диффи-Хеллмана работает только на линиях связи, надёжно защищённых от модификации. Если бы он был применим на любых открытых каналах, то давно снял бы проблему распространения ключей и, возможно, заменил бы собой всю асимметричную криптографию.

Пример. Пусть $n = 97$ и $g = 5$. Абонент A сгенерировал случайное число $X_A = 36$. Абонент B сгенерировал случайное число $X_\text{B} = 58$. Эти элементы они держат в секрете. Далее каждый из них вычисляет новый элемент:

$$Y_A = 5^{36} \bmod 97 = 50, \quad Y_B = 5^{58} \bmod 97 = 44.$$

Затем они обмениваются этими элементами по каналу связи. Теперь абонент A, получив Y_B и зная свой секретный элемент X_A, вычисляет общий ключ: $K_A = 44^{36} \bmod 97 = 75$. Аналогично поступает абонент B: $K_B = 50^{58} \bmod 97 = 75$.

Задание

Примечание: В данной лабораторной работе под большими числами будем понимать числа, превышающие 2^{64}.

I. Реализовать приложение, позволяющее выполнять следующие действия:
   1. Генерировать большие простые числа:
      1) программа по заданным t (количество проверок в тесте Рабина-Миллера) и n (количество бит) должна генерировать простое n-битное число, отображая при этом, сколько итераций алгоритма генерации простого числа потребовалось выполнить для его генерации и сколько времени было затрачено на это;
      2) программа по заданным границам диапазона должна выводить все простые числа из этого диапазона, отображая время, затраченное на генерацию всех чисел.
   2. Определять для заданного числа первые 100 первообразных корней, отображая при этом суммарное время, затраченное программой на их поиск.
   3. Моделировать обмен ключами между абонентами по схеме Диффи-Хеллмана. Программа должна получать большие простые числа X_A , X_B и n случайным образом с помощью алгоритма генерации простого числа, а также предоставлять пользователю возможность задавать их.

II. С помощью реализованного приложения выполнить следующие задания:
   1. Протестировать правильность работы разработанного приложения.
   2. Сделать выводы о проделанной работе.

Дополнительные критерии оценивания качества работы

1. Отображение программой числа итераций и затраченного времени:
   1 – программа отображает и число итераций, и затраченное на генерацию время;
   0 – иначе.

Вопросы для защиты

I. **Первая часть защиты** (обязательная):
   1. Для чего нужно большое простое число? Как проверить, является число простым или нет? Как сгенерировать большое простое число?
   2. Алгоритм эффективной реализации возведения целого числа в целую степень по модулю n.
   3. На чём основывается безопасность обмена ключа по схеме Диффи-Хеллмана?
   4. Как происходит обмен ключами по схеме Диффи-Хеллмана?
   5. Доказать, что в схеме Диффи-Хеллмана $K_A = K_B$.

II. **Вторая часть защиты**: Для заданного n найти функцию Эйлера $\varphi(n)$.
   1. $n = 11$;
   2. $n = 6$;
   3. $n = 21$;
   4. $n = 10$;
   5. $n = 105$;
   6. $n = 18$.

III. **Третья часть защиты**: Для заданного n найти все первообразные корни.
   1. $n = 6$;
   2. $n = 7$;
   3. $n = 9$;
   4. $n = 11$;
   5. $n = 13$;
   6. $n = 19$.

Список литературы

1. Харин, Ю.С. Математические и компьютерные основы криптологии : учебное пособие / Ю.С. Харин, В.И. Берник, Г.В. Матвеев, С.В. Агиевич. – Мн. : Новое знание, 2003. – 382 с.
2. Шнайер, Б. Прикладная криптография. Протоколы, алгоритмы, исходные тексты на языке Си / Б. Шнайер. – М. : Триумф, 2002. – 816 с.
3. Rescorla, E. Diffie-Hellman Key Agreement Method : RFC 2631 / E. Rescorla ; RTFM Inc. – CA 94303 : Network Working Group, 1999. – 13 p.