Задача № 1
Реализация сетевого протокола выработки секретного ключа

Протокол Диффи–Хеллмана (Diffie, Hellman)

Исходно DH-протокол описывался в рамках простого поля и его мультипликативной группы. Пусть $\mathbb{Z}_p$ – простое поле (p – простое число), $\mathbb{Z}_p^*$ – его мультипликативная группа с генератором g (числа p и g несекретны и могут коллективно использоваться сообществом). Пользователи A и B выбирают секретные ключи соответственно $x, y \in \mathbb{Z}_p$, вычисляют открытые ключи $X = g^x$, $Y = g^y$ и обмениваются открытыми ключами (по открытому каналу). Общий для A и B ключ вычисляется ими как $Y^x = X^y$.

По состоянию знаний на сегодняшний день предпочтительна реализация протокола, когда открытые ключи являются элементами не мультипликативной группы $\mathbb{Z}_p^*$, а циклической подгруппы *G* простого порядка q. В этом случае в качестве g берется генератор такой подгруппы, секретные ключи $x, y \in \mathbb{Z}_q$, все вычисления выполняются точно так же. Длины чисел, при которых обеспечивается долговременная стойкость по отношению к известным алгоритмам взлома, $|p| = 1024$ бит, $|q| = 160$ бит (в США) или 256 бит (в РФ). Так как вычисленный общий ключ принадлежит *G*, но выражен числом длиной $|p|$ бит, его рекомендуется "сжать" до $|q|$ бит с помощью криптографической хэш-функции H. То есть результирующий ключ вычисляется как $K = H(Y^x) = H(X^y)$. Хэш-функция также предотвращает манипуляцию битами ключа *K* путём неслучайного выбора чисел x и y.

Отметим, что подгруппа *G* может выбираться не только в рамках простого поля $\mathbb{Z}_p$, но и в других алгебраических структурах, например, в бинарном поле $\mathbb{F}_{2^m}$ или в группе точек на эллиптической кривой.

Протокол MQV (Menezes, Qu, Vanstone)

Пусть опять имеется циклическая подгруппы *G* простого порядка q с генератором g. Обозначим через l половину битовой длины числа q: $l = |q|/2$. Пользователи A и B генерируют долговременные пары секретных и открытых ключей соответственно a, A и b, B: $a, b \in \mathbb{Z}_q$ (выбираются случайным образом), $A = g^a$, $B = g^b$. Открытые ключи пересылаются всем пользователям в виде сертифицированного (подписанного удостоверяющим центром) справочника.

Для построения общего сеансового ключа пользователь A генерирует случайное число $x \in \mathbb{Z}_q$ и вычисляет $X = g^x$. Пользователь B генерирует случайное число $y \in \mathbb{Z}_q$ и вычисляет $Y = g^y$. Затем пользователи обмениваются числами X и Y по открытому каналу связи. Оба пользователя вычисляют числа $d = 2^l + (X \bmod 2^l)$, $e = 2^l + (Y \bmod 2^l)$. Пользователь A вычисляет $S_\mathrm{A} = (YB^e)^{x+da}$. Пользователь B вычисляет $S_\mathrm{B} = (XA^d)^{y+eb}$ (напомним, что при возведении в степень показатели приводятся $\mathrm{mod}\ q$). Результирующий ключ получается следующим образом: $K = H(S_\mathrm{A}) = H(S_\mathrm{B})$.

Измерение времени вычислений с помощью счетчика циклов процессора

Если используется компилятор GCC, необходимо подключить файл `rdtsc.h`. В этом файле определены две функции:

```
unsigned long long rdtsc ()
```

возвращает полное (64 бита) текущее значение счетчика циклов процессора.

```
unsigned int CC ()
```

возвращает младшее слово (32 бита) счетчика циклов процессора, что достаточно, если заранее известно, что измеряемое время не превышает 2^{32} циклов процессора.

Задание

1. Самостоятельно изучить библиотеку gmp для реализации арифметики с длинными числами. Руководство находится в файле `GMP/gmp-man-6.1.0.pdf`. В этом же каталоге есть пример программы с использованием функций gmp (проект для Dev-CPP – `gmp.dev`, включающий один файл `gmp.cpp`). Файлы `gmp.h`, `libgmp.a`, `libgma.la` можно взять для установки библиотеки на своём компьютере. По желанию студента допускается использовать другие известные ему средства реализации арифметики с длинными числами.

2. Реализовать программу генерации чисел p, q, g для операций в мультипликативной группе $\mathbb{Z}_p^*$ и в циклической подгруппе G порядка q. Сгенерированные числа сохранить для последующего использования в файле.

3. Реализовать исходный алгоритм Диффи–Хеллмана, алгоритм Диффи–Хеллмана в подгруппе, алгоритм MQV. Пока без хэш-функций на последнем этапе.

4. Сделать сетевые версии программ, реализующие действия пользователей A и B на разных компьютерах локальной сети.

5. Осуществить замеры времени (в виде числа процессорных циклов) при выполнении основных действий во всех алгоритмах и провести их сопоставление.