МИНИСТЕРСТВО ПО РАЗВИТИЮ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И КОММУНИКАЦИЙ РЕСПУБЛИКИ УЗБЕКИСТАН

ТАШКЕНТСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ ИМЕНИ МУХАММАДА АЛ-ХОРАЗМИЙ

ФАКУЛЬТЕТ ИНФОРМАЦИОННАЯ БЕЗОПАСНОСТЬ

**Кафедра Криптология**

# Методическое пособие по выполнению практических работ учебной дисциплины «Криптография II» для студентов направлении информационной безопасности

**Ташкент - 2021**

**Авторы:** Преподаватели кафедры "Криптологии" Турсунов О.О., Ахмедова Н.Ф., Исломов Ш.З. Методическое пособие по выполнению практических работ по учебной дисциплине «Криптография II» для студентов бакалавриатуры специальностей информационной безопасности - Ташкент: ТУИТ. 2021.-68 с.

Методическое пособие по выполнению практических работ по дисциплине «Криптография II» является продолжением дисциплины «Криптография I» и содержит материалы по следующим темам: шифрование данных по алгоритму RSA, метод факторизации Ферма, дискретное логарифмирование, шифрование по алгоритму Эль-Гамаля, эллиптическая криптография, алгоритм Рабина, шифрование данных гибридным методом, разработка программных модулей данных алгоритмов, а также применение библиотеки OpenSSL для шифрования.

Методическое пособие для студентов Ташкентского университета информационных технологий имени Мухаммада аль-Хорезми. Руководство рекомендовано к публикации решением Научно-методического совета Ташкентского университета информационных технологий имени Мухаммада аль-Хорезми (протокол № «10(135)» май » «25» 2021 г.

**Практическая работа № 1**

**Тема:** Шифрование данных по алгоритму RSA с использованием библиотеки OpenSSL

**Цель работы:** Формирование знаний и навыков шифрования данных по алгоритму RSA с помощью библиотеки OpenSSL.

**Теоретическая часть**

*Алгоритм RSA*

RSA (аббревиатура от фамилий Rivest, Shamir и Adleman) — криптографический алгоритм с открытым ключом, основывающийся на вычислительной сложности задачи факторизации больших целых чисел.

Зашифруем и расшифруем сообщение "CAB" по алгоритму RSA. Для простоты возьмем небольшие числа - это сократит наши расчеты.

- Выберем p=3 and q=11.
- Определим n= 3*11=33.
- Найдем (p-1)*(q-1)=20. Следовательно, *e* будет равно, например, 7: (*e*=7).
- Выберем число *d* по следующей формуле: (*d**7) mod 20=1. Значит е будет равно, например, 3: (*d*=3).
- Представим шифруемое сообщение как последовательность чисел в диапозоне от 0 до 32 (незабывайте, что кончается на n-1). Буква A =1, B=2, C=3.

*Теперь зашифруем сообщение, используя открытый ключ {7,33}*

C1 = (3^7) mod 33 = 2187 mod 33 = 9;

C2 = (1^7) mod 33 = 1 mod 33 = 1;

C3 = (2^7) mod 33 = 128 mod 33 = 29;

*Теперь расшифруем данные, используя закрытый ключ {3,33}.*

M1=(9^3) mod 33 =729 mod 33 = 3(C);

M2=(1^3) mod 33 =1 mod 33 = 1(A);

M3=(29^3) mod 33 = 24389 mod 33 = 2(B);

## Практическая часть

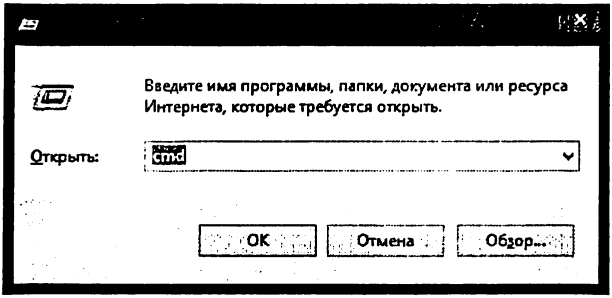



Рис.1.1 Результат комбинации клавиш Win+R

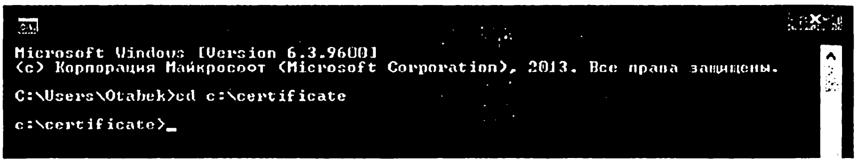



Рис.1.2. Путь к папке certificate через командную строку

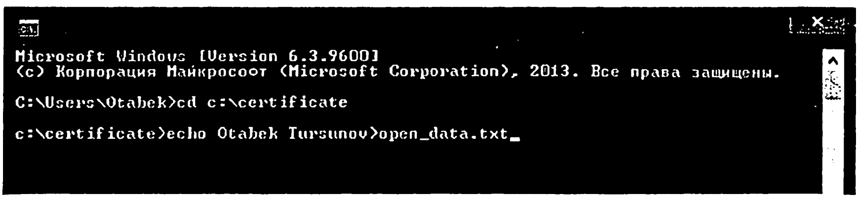



Рис.1.3. Введение команды открытия файла

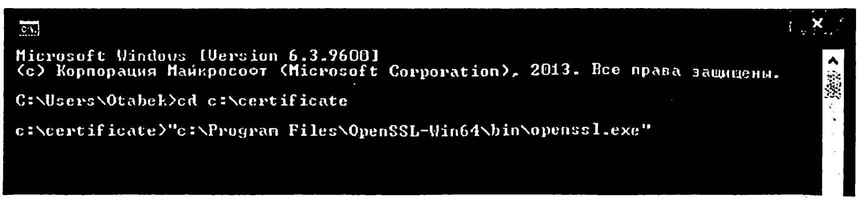



Рис.1.4. Перемещение openssl.exe с указанного места в каталог certificate

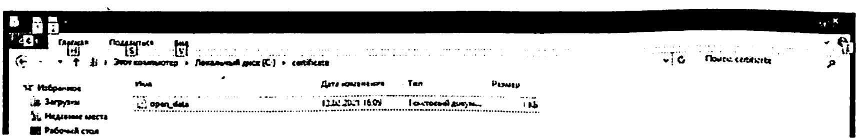


Рис.1.5. Окно каталога certificate

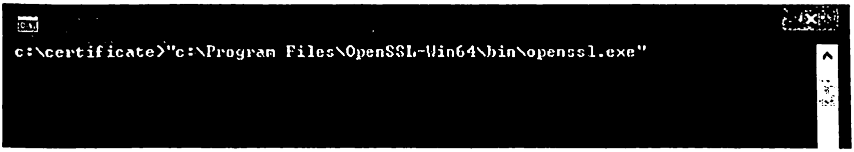



Рис.1.6. Открытие Openssl

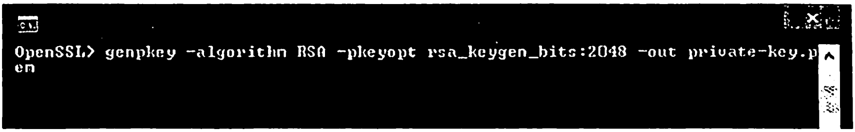



Рис.1.7. Генерация личного ключа

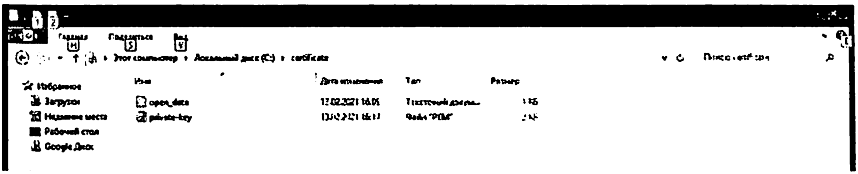



Рис.1.8. Сгенерированный личный ключ в каталоге certificate

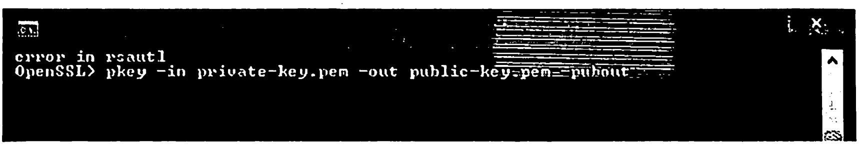



Рис.1.9. Генерация открытого ключа

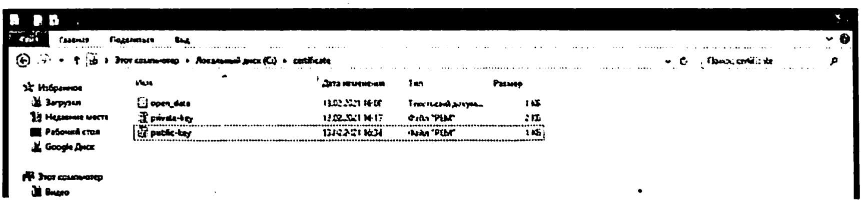



Рис.1.10. Сгенерированный ключ в каталоге certificate

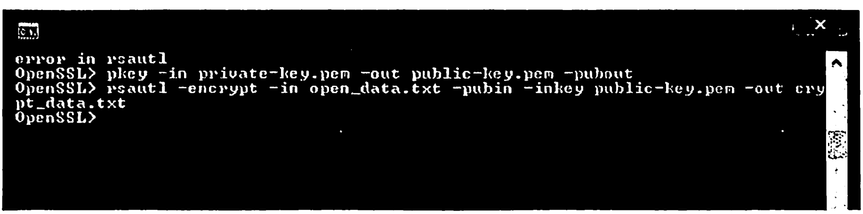



Рис.1.11. Шифрование данных

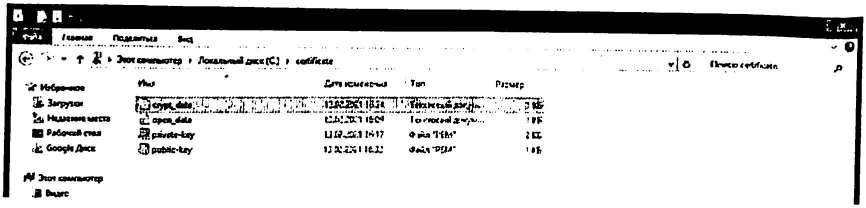



Рис.1.12. Зашифрованные данные в виде файла в каталоге

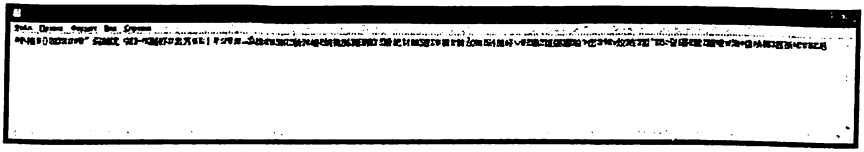


Рис.1.13. Содержание crypt_data

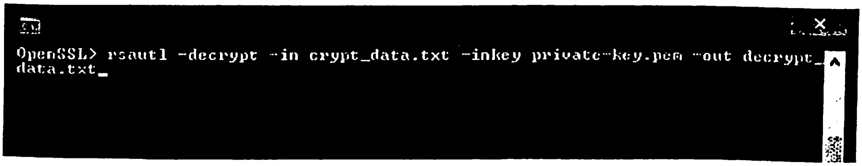



Рис.1.14. Расшифрование данных

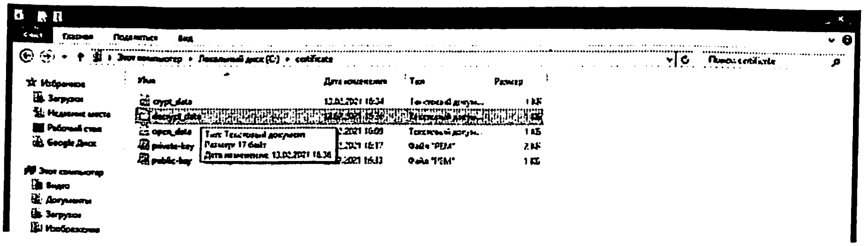



Рис.1.15. Расшифрованные данные в виде файла в каталоге

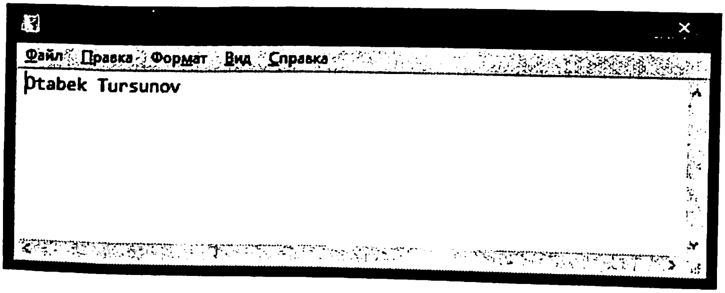



Рис.1.16. Содержание файла decrypt_data

**Задание**

Зашифровать своё имя с помощью следующих значений по алгоритму RSA.

**Варианты:**

| № | p | q |
|---|---|---|
| **1.** | 19 | 11 |
| **2.** | 17 | 7 |
| **3.** | 23 | 13 |
| **4.** | 29 | 5 |
| **5.** | 19 | 7 |
| **6.** | 5 | 17 |
| **7.** | 13 | 11 |
| **8.** | 31 | 7 |
| **9.** | 29 | 11 |
| **10.** | 29 | 17 |
| **11.** | 19 | 31 |
| **12.** | 13 | 17 |
| **13.** | 23 | 19 |
| **14.** | 7 | 17 |
| **15.** | 11 | 37 |
| **16.** | 13 | 23 |
| **17.** | 23 | 37 |
| **18.** | 37 | 7 |
| **19.** | 7 | 29 |
| **20.** | 11 | 13 |
| **21.** | 23 | 5 |
| **22.** | 29 | 7 |
| **23.** | 31 | 19 |

## Практическая работа № 2

**Тема:** Разработка программного средства, решающее проблему факторизации

**Цель работы:** Формирование знаний и навыков по алгоритму Ферма и Ро-алгоритму, а также по разработке программного средства по этим алгоритмам.

### Теоретическая часть

*Метод факторизации Ферма*

Метод факторизации Ферма — алгоритм факторизации (разложения на множители) нечётного целого числа *n*, предложенный Пьером Ферма (1601—1665) в 1643 году.

*Описание алгоритма*

Метод основан на поиске таких целых чисел $x$ и $y$, которые удовлетворяют соотношению $x^2 - y^2 = n$, что ведёт к разложению *n=(x-y)*(x+y)*.

Для разложения на множители нечётного числа *n* ищется пара чисел *(x,y)* таких, что $x^2 - y^2 = n$, или *(x-y)*(x+y)=n*. При этом числа *(x+y)* и *(x-y)* являются делителями *n*, возможно, тривиальными (то есть одно из них равно 1, а другое - *n*.)

В нетривиальном случае, равенство $x^2 - y^2 = n$ равносильно $x^2 - n = y^2$, то есть тому, что $x^2 - n$ является квадратом.

Поиск квадрата такого вида начинается с $x = \sqrt{n}$ - наименьшего числа, при котором разность $x = \sqrt{n}$ неотрицательна.

Для каждого значения $k \in N$ начиная с *k=1*, вычисляют $([\sqrt{n}] + k)^2 - n$ и проверяют, не является ли это число точным квадратом. Если не является, то $k$ увеличивают на единицу и переходят на следующую итерацию.

Если $([\sqrt{n}] + k)^2 - n$ является точным квадратом, то есть $x^2 - n = ([\sqrt{n}] + k)^2 - n = y^2$, то получено разложение:

$n = x^2 - y^2 = (x - y) * (x + y) = a * b$, в котором $x = [\sqrt{n}] + k)$

Если оно является тривиальным и единственным, *n* - простое.

На практике значение выражения на *(k+1)* - ом шаге вычисляется с учётом значения на *k*-ом шаге:

$$(s+1)^2 - n = s^2 + 2 * s + 1 - n, \text{ где } s = [\sqrt{n}] + k.$$

*Пример с малым числом итераций*

Возьмём число *n=10873*. Вычислим $s = [\sqrt{n}] = 105$. Для $k = 0,1,2,\ldots$ будем вычислять значения функции *s+k*. Для дальнейшей простоты построим таблицу, которая будет содержать значения $y = (s+k)^2 - n$ и $\sqrt{y}$ на каждом шаге итерации. Получим:

| $k$ | $y$ | $\sqrt{y}$ |
|---|---|---|
| 1 | 363 | 19.052 |
| 2 | 576 | 24 |

Как видно из таблицы, уже на втором шаге итерации было получено целое значение $\sqrt{y}$.

Таким образом имеет место следующее выражение: $(105 + 2)^2 - n = 24^2$. Отсюда следует, что $n = 107^2 - 24^2 = 131 * 83$.

*Пример с большим числом итераций*

Пусть $n = 89755$. Тогда $\sqrt{n} \approx 299.591$ или $s = [\sqrt{n}] = 300$

| $k$ | $y$ | $\sqrt{y}$ |
|---|---|---|
| 77 | 52374 | 228.854 |
| 78 | 53129 | 230.497 |
| 79 | 53886 | 232.134 |
| 80 | 54645 | 233.763 |
| 81 | 55406 | 235.385 |
| 82 | 56139 | 237 |

$\sqrt{y} = 237$

$a = s + k + \sqrt{y} = 300 + 82 + 237 = 619$

$b = s + k - \sqrt{y} = 300 + 82 - 237 = 145$

Данное разложение является не конечным, так как, очевидно, что число 145 не является простым: 145=29 * 5.

В итоге, конечное разложение исходного числа *n* на произведение простых множителей 89755=5 * 29 * 619.

*Ро-алгоритм*

Ро-алгоритм ($\rho$-алгоритм) — предложенный Джоном Поллардом в 1975 году алгоритм, служащий для факторизации (разложения на множители) целых чисел. Данный алгоритм основывается на алгоритме Флойда поиска длины цикла в последовательности и некоторых следствиях из парадокса дней рождения. Алгоритм наиболее эффективен при факторизации составных чисел с достаточно малыми множителями в разложении. Сложность алгоритма оценивается как.

$\rho$-алгоритм Полларда строит числовую последовательность, элементы которой образуют цикл, начиная с некоторого номера n, что может быть проиллюстрировано, расположением чисел в виде греческой буквы $\rho$, что послужило названием семейству алгоритмов.

*Описание алгоритма*

Данный пример наглядно демонстрирует ρ-алгоритм факторизации (версия алгоритма, с улучшением Флойда), для числа $N = 8051$:

Таблица: факторизация числа 8051

| $n = 8051, F(x) = (x^2 + 1) \bmod n, x_0 = y_0 = 2$ | | | |
|---|---|---|---|
| $i$ | $x_i = F(x_{i-1})$ | $y_i = F(F(y_{i-1}))$ | НОД($\lvert x_i - y_i \rvert$, 8051) |
| 1 | 5 | 26 | 1 |
| 2 | 26 | 7474 | 1 |
| 3 | 677 | 871 | 97 |

Используя другие варианты полинома $F(x)$, можно также получить делитель 83:

Таблица: факторизация числа 8051

| $n = 8051, F(x) = (x^2 + 3) \bmod n, x_0 = y_0 = 2$ | | | |
|---|---|---|---|
| $i$ | $x_i=F(x_{i-1})$ | $y_i=F(F(y_{i-1}))$ | НОД($\lvert x_i - y_i \rvert$, 8051) |
| 1 | 7 | 52 | 1 |
| 2 | 52 | 1442 | 1 |
| 3 | 2707 | 778 | 1 |
| 4 | 1442 | 3932 | 83 |

Таким образом, $d_1 = 97$, $d_2 = 83$ — нетривиальные делители числа 8051.

После нахождения делителя числа, в ρ-алгоритме предлагается продолжать вычисления и искать делители числа $N/d$, если $N/d$ не является простым. В этом простом примере данного шага совершать не потребовалось.

**Задание**

Разложите на множители с помощью алгоритмов ферма и $\rho$-алгоритм Полларда.

| № | N |
|---|---|
| 1. | 5893 |
| 2. | 3393 |
| 3. | 3901 |
| 4. | 4797 |
| 5. | 7571 |
| 6. | 5031 |
| 7. | 8137 |
| 8. | 6499 |
| 9. | 6157 |
| 10. | 7979 |
| 11. | 4453 |
| 12. | 2173 |
| 13. | 3977 |
| 14. | 2059 |
| 15. | 5723 |
| 16. | 5561 |
| 17. | 4559 |
| 18. | 4181 |

| | |
|---|---|
| **19.** | 3277 |
| **20.** | 5311 |
| **21.** | 3007 |
| **22.** | 3277 |
| **23.** | 4189 |

## Практическая работа № 3

**Тема:** Разработка программного средства, решающее задачу дискретного логарифмирования

**Цель работы:** Формирование знаний и навыков по разработке программного средства и решению задач по алгоритму Полига-Хеллмана.

### Теоретическая часть

*Дискретное логарифмирование* (DLOG) — задача обращения функции $g^x$ в некоторой конечной мультипликативной группе $G$.

Наиболее часто задачу дискретного логарифмирования рассматривают в мультипликативной группе кольца вычетов или конечного поля, а также в группе точек эллиптической кривой над конечным полем. Эффективные алгоритмы для решения задачи дискретного логарифмирования в общем случае неизвестны.

Для заданных $g$ и а решение $x$ уравнения $g^x = a$ называется *дискретным логарифмом* элемента а по основанию $g$. В случае, когда $G$ является мультипликативной группой кольца вычетов по модулю $m$, решение называют также индексом числа $a$ по основанию $g$. Индекс числа а по основанию $g$ гарантированно существует, если $g$ является первообразным корнем по модулю $m$.

*Пример*

Пусть в некоторой конечной мультипликативной абелевой группе $G$ задано уравнение $g^x$.

Решение задачи дискретного логарифмирования состоит в нахождении некоторого целого неотрицательного числа $x$, удовлетворяющего уравнению. Если оно разрешимо, у него должно быть хотя бы одно натуральное решение, не превышающее порядок группы. Это сразу даёт грубую оценку сложности алгоритма поиска решений сверху — алгоритм полного перебора нашёл бы решение за число шагов не выше порядка данной группы.

Чаще всего рассматривается случай, когда $G = (g)$, то есть группа является циклической, порождённой элементом $g$. В этом случае уравнение всегда имеет решение. В случае же произвольной группы вопрос о разрешимости задачи дискретного логарифмирования, то есть вопрос о существовании решений уравнения, требует отдельного рассмотрения.

Рассмотрим задачу дискретного логарифмирования в кольце вычетов по модулю простого числа.

Пусть задано сравнение. $3^x \equiv 13 \ (mod\ 17)$

Будем решать задачу методом перебора. Выпишем таблицу всех степеней числа 3. Каждый раз мы вычисляем остаток от деления на 17 (например, 33≡27 — остаток от деления на 17 равен 10).

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| $3^1 \equiv 3$ | $3^2 \equiv 9$ | $3^3 \equiv 10$ | $3^4 \equiv 13$ | $3^5 \equiv 5$ | $3^6 \equiv 15$ | $3^7 \equiv 11$ | $3^8 \equiv 16$ |
| $3^9 \equiv 14$ | $3^{10} \equiv 8$ | $3^{11} \equiv 7$ | $3^{12} \equiv 4$ | $3^{13} \equiv 12$ | $3^{14} \equiv 2$ | $3^{15} \equiv 6$ | $3^{16} \equiv 1$ |

Теперь легко увидеть, что решением рассматриваемого сравнения является $x = 4$, поскольку 3^4≡13.

На практике модуль обычно является достаточно большим числом, и метод перебора является слишком медленным, поэтому возникает потребность в более быстрых алгоритмах.

*Алгоритм Полига - Хеллмана*

Алгоритм Полига - Хеллмана (также называемый алгоритм Сильвера - Полига - Хеллмана) - детерминированный алгоритм дискретного логарифмирования в кольце вычетов по модулю простого числа. Одной из особенностей алгоритма является то, что для простых чисел специального вида можно находить дискретный логарифм за полиномиальное время.

Суть алгоритма в том, что достаточно найти $x$ по модулям $q^{a_i}{}_i$ для всех $i$, а затем решение исходного сравнения можно найти с помощью китайской теореме об остатках. Чтобы найти $x$ по каждому из таких модулей, нужно решить сравнение:

$$a^{q_i{}^{\frac{p-1}{a_i}}} \equiv b^{q_i{}^{\frac{p-1}{a_i}}} mod\, p$$

Данное сравнение решается за полиномиальное время в случае, если $q_i$ - небольшое (то есть, не превосходит $log\, p^c$, где $c$ – некоторая константа).

Описание алгоритма

*Упрощённый вариант*

Лучшим путём, чтобы разобраться с данным алгоритмом, будет рассмотрение особого случая, в котором $p = 2^{\tau} + 1$ .

Нам даны $a, P, b$, при этом $a$ есть примитивный элемент $G\,F(p)$ и нужно найти такое $x$, чтобы удовлетворялось $a^x \equiv b\ (mod\,p)$.

Принимается, что $0 \le x \le p - 2$, так как $x = p - 1$ неотличимо от $x = 0$, потому что в нашем случае примитивный элемент $a$ по определению имеет степень *p-1*, следовательно:

$$a^{p-1} \equiv 1 \equiv a^0\ (mod\,p).$$

Когда $p = 2^n + 1$, легко определить $x$ двоичным разложением с коэффициентами $\{q_0, q_1, \dots, q_{n-1}\}$, например:

$$x = \sum_{i=0}^{n-1} q_i 2^i = q_0 + q_1 2^1 + \cdots + q_{n-1} 2^{n-1}$$

Самый младший бит $q_0$ определяется путём возведения $b$ в степень $(p - 1)/2 = 2^{n-1}$ и применением правила

$$b^{(p-1)/2} \pmod p \equiv \begin{cases} +1, & q_0 = 0 \\ -1, & q_0 = 1. \end{cases}$$

*Вывод верхнего правила*

Рассмотрим ранее полученное сравнение:

$$a^{p-1} \equiv 1\ (mod\,p) \Rightarrow a^{\frac{-1}{2}} \equiv \pm\ (mod\,p),$$

Но $a$ в степени *(p-1)/2 < p-1* по определению принимает значение, отличное от *1*, поэтому остаётся только одно сравнение:

$$a^{(p-1)/2} \equiv 1 \equiv -1\ (mod\,p).$$

Возведём сравнение (1) в степень *(p-1)/2* и подставим выше полученное сравнение:

$$b^{(p-1)/2} \equiv (a^x)^{(p-1)/2} \equiv (a^{(p-1)/2})^x \equiv (-1)^x \pmod p$$

Равенство $-1^x = 1$ верно, если $x$ чётное, то есть в разложении в виде многочлена свободный член $q_0$ равен 0, следовательно $-1^x = -1$ верно, когда $q_0 = 1$.

Теперь преобразуем известное разложение и введём новую переменную $z_1$:

$$b \equiv a^x \equiv a^{x_1+q_0} \pmod p \Rightarrow z_1 \equiv ba^{-q_0} \equiv a^{x_1} \pmod p,$$

где

$$x_1 = \sum_{i=1}^{n-1} q_i 2^i = q_1 2^1 + q_2 2^2 + \cdots + q_{n-1} 2^{n-1}$$

Понятно, что $x_1$ делится на 4 при $q_1 = 0$, а при $q_1 = 1$ делится на 2, а на 4 уже нет.

Рассуждая как раньше, получим сравнение:

$$z_1^{(p-1)/4} \pmod p \equiv \begin{cases} +1, & q_1 = 0 \\ -1, & q_1 = 1, \end{cases}$$

из которого находим $q_1$.

Оставшиеся биты получаются похожим способом. Напишем общее решение нахождения $q_i$ с новыми обозначениями:

$$m_i = (p-1)/2^{i+1}$$

$$z_i \equiv b \cdot a^{-q_0 - q_1 2^1 - \ldots - q_{i-1} 2^{i-1}} \equiv a^{x_i} \pmod p,$$

где

$$x_i = \sum_{k=i}^{n-1} q_k 2^k.$$

Таким образом, возведение $z_i$ в степень $m_i$ даёт:

$$z_i^{m_i} \equiv a^{(x_i \cdot m_i)} \equiv \left(a^{(p-1)/2}\right)^{(x_i/2^i)} \equiv (-1)^{x_i/2^i} \equiv (-1)^{q_i} \pmod p.$$

Следовательно:

$$z_i^{m_i} \pmod p \equiv \begin{cases} +1, & q_i = 0 \\ -1, & q_i = 1, \end{cases}$$

из которого находим $q_i$.

Найдя все биты, получаем требуемое решение $x$.

## Практическая часть

*Пример 1*

Дано:

$$a = 3,\ b = 11,\ p = 17 = 2^4 + 1$$

Найти: $x$

Решение:

Получаем $p - 1 = 2^4$. Следовательно $x$ имеет вид:

$$x = q_0 + q_1 2^1 + q_2 2^2 + q_3 2^3$$

Находим $q_0$:

$$b^{(p-1)/2} \equiv 11^{(17-1)/2} \equiv 11^8 \equiv (-6)^8 \equiv (36)^4 \equiv 2^4 \equiv 16 \equiv -1 \pmod{17} \Rightarrow q_0 = 1$$

Подсчитываем $z_i$ и $m_i$:

$$z_1 \equiv b \cdot a^{-q_0} \equiv 11 \cdot 3^{-1} \equiv 11 \cdot 6 \equiv 66 \equiv -2 \pmod{17}$$

$$m_1 = (p-1)/2^{1+1} = (17-1)/2^2 = 4$$

Находим $q_1$:

$$z_1^{m_1} \equiv (-2)^4 \equiv 16 \equiv -1 \pmod{17} \Rightarrow q_1 = 1$$

Подсчитываем $z_2$ и $m_2$:

$$z_2 \equiv z_1 \cdot a^{-q_1 2^1} \equiv (-2) \cdot 3^{-2} \equiv (-2) \cdot 6^2 \equiv (-2) \cdot 36 \equiv (-2) \cdot 2 \equiv -4 \equiv 13 \pmod{17}$$

$$m_2 = (p-1)/2^{2+1} = (17-1)/2^3 = 2$$

Находим $q_i$:

$$z_2^{m_2} \equiv 13^2 \equiv (-4)^2 \equiv 16 \equiv -1 \pmod{17} \Rightarrow q_2 = 1$$

Подсчитываем $z_3$ и $m_3$:

$$z_3 \equiv z_2 \cdot a^{-q_2 \cdot 2^2} \equiv 13 \cdot 3^{-4} \equiv 13 \cdot 9^{-2} \equiv 13 \cdot 2^2 \equiv (-4) \cdot 4 \equiv -16 \equiv 1 \pmod{17}$$

$$m_3 = (p-1)/2^{3+1} = (17-1)/2^4 = 1$$

Находим $q_3$:

$$z_3^{m_3} \equiv 1^1 \equiv 1 \pmod{17} \Rightarrow q_3 = 0$$

Находим искомый $x$: $x = 1 + 1 \cdot 2^1 + 1 \cdot 2^2 + 0 \cdot 2^3 \equiv 7$

Ответ: $x = 7$

**Задание**

Найдите $x$ с помощью алгоритма Полига - Хеллмана.

**Варианты:**

| **№** | **Уравнение** |
|---|---|
| **1.** | $3^x \equiv 26 \mod 39$ |
| **2.** | $7^x \equiv 18 \mod 41$ |
| **3.** | $5^x \equiv 11 \mod 39$ |
| **4.** | $8^x \equiv 19 \mod 43$ |
| **5.** | $16^x \equiv 14 \mod 53$ |
| **6.** | $11^x \equiv 24 \mod 37$ |
| **7.** | $7^x \equiv 16 \mod 37$ |
| **8.** | $3^x \equiv 26 \mod 53$ |
| **9.** | $7^x \equiv 11 \mod 19$ |
| **10.** | $8^x \equiv 11 \mod 29$ |
| **11.** | $13^x \equiv 7 \mod 43$ |
| **12.** | $9^x \equiv 19 \mod 53$ |
| **13.** | $15^x \equiv 5 \mod 37$ |
| **14.** | $7^x \equiv 29 \mod 37$ |
| **15.** | $7^x \equiv 12 \mod 17$ |
| **16.** | $11^x \equiv 13 \mod 23$ |
| **17.** | $13^x \equiv 35 \mod 43$ |
| **18.** | $9^x \equiv 41 \mod 53$ |
| **19.** | $10^x \equiv 36 \mod 41$ |
| **20.** | $15^x \equiv 32 \mod 37$ |
| **21.** | $18^x \equiv 14 \mod 23$ |
| **22.** | $10^x \equiv 13 \mod 29$ |
| **23.** | $9^x \equiv 22 \mod 37$ |

## Практическая работа № 4

**Тема:** Разработка программного средства для шифрования данных по алгоритму Эль-Гамаля

**Цель работы:** Формирование знаний и навыков по разработке программного средства и решению задач по алгоритму Эль-Гамаля.

### Теоретическая часть

*Схема Эль-Гамаля* (Elgamal) — криптосистема с открытым ключом, основанная на трудности вычисления дискретных логарифмов в конечном поле. Криптосистема включает в себя алгоритм шифрования и алгоритм цифровой подписи. Схема Эль-Гамаля лежит в основе бывших стандартов электронной цифровой подписи в США (DSA) и России (ГОСТ Р 34.10-94).

Схема была предложена Тахером Эль-Гамалем в 1985 году. Эль-Гамаль разработал один из вариантов алгоритма Диффи-Хеллмана. Он усовершенствовал систему Диффи-Хеллмана и получил два алгоритма, которые использовались для шифрования и для обеспечения аутентификации. В отличие от RSA, алгоритм Эль-Гамаля не был запатентован и поэтому стал более дешёвой альтернативой, так как не требовалась оплата взносов за лицензию. Считается, что алгоритм попадает под действие патента Диффи-Хеллмана.

*Генерация ключей*

Первый этап алгоритма Эль-Гамаля заключается в генерации ключей. Этот этап включает следующую последовательность действий:

1. Генерируется случайное простое число $p$ длины $n$ бит.

2. Выбирается произвольное целое число $g$, являющееся *первообразным (примитивным) корнем* по модулю $p$.

3. Выбирается случайное число $x$ из интервала $1<x<p-1$, взаимно простое с $p-1$.

4. Вычисляется $y = g^x (mod\, p)$.

5. *Открытым ключом* является тройка $(g, p, y)$, *закрытым ключом* — число $x$.

*Работа в режиме шифрования*

Второй этап алгоритма включает *шифрование и расшифрование.*

*Шифрование*

Сообщение $M$ должно быть меньше числа $p$. Сообщение шифруется следующим образом:

1. Выбирается случайное секретное число $k$ из интервала $1<k<p-1$, взаимно простое с $p-1$.

2. Вычисляется $a = g^k (mod\,p)$, $b = M * y^k (mod\,p)$, где $M$ — исходное сообщение. Пара чисел $(a, b)$ является *шифртекстом.*

*Расшифрование*

Зная закрытый ключ $x$ и учитывая тот факт, что $M = (b * a^{p-1-x}) (mod\,p)$, исходное сообщение $M$ можно вычислить из шифртекста $(a, b)$ по формуле:

$M = a^{-x} * b\ (mod\,p)$

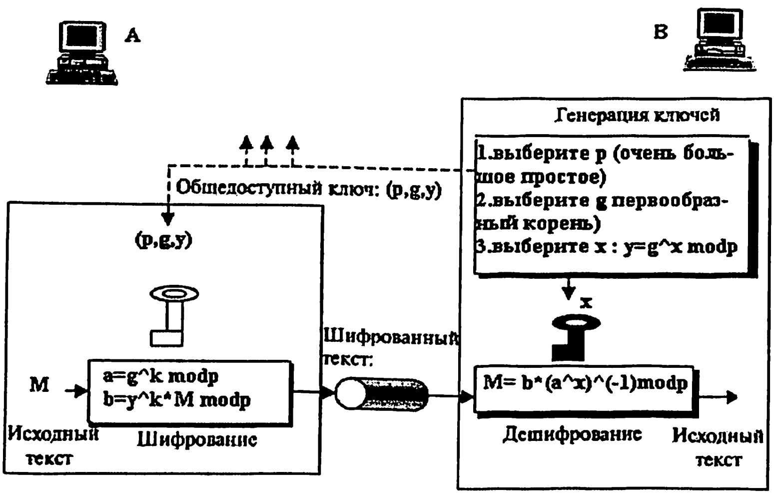



Рис. 4.1. Схема шифрования алгоритма Эль-Гамаля

Суть задачи заключается в следующем. Имеется уравнение $g^x\ mod\,p = y$. Требуется по известным $g$, $y$ и $p$ найти целое неотрицательное число **x** **(дискретный логарифм).**

Порядок создания ключей приводится в следующей таблице.

Таблица 4.1. Процедура создания ключей

| № п/п | Описание операции | Пример |
|---|---|---|
| 1 | Выбирается простое число p. | p=37 |
| 2 | Выбирается число g, являющееся первообразным корнем по модулю p и меньшее p. | g=2 |
| 3 | Выбираются произвольное число x, меньшее p. | x=5 |
| 4 | Вычисляется $y = g^x \bmod p$ | $y = 2^5 \bmod 37 = 32 \bmod 37 = 32$ |
| 5 | Открытый ключ - y, g и p. Причем g и p можно сделать общими для группы пользователей. Закрытый ключ - x. | |

Для шифрования каждого отдельного блока исходного сообщения должно выбираться случайное число *k* ($1 < k < p - 1$). После чего шифрограмма генерируется по следующим формулам $a = g^k \; mod \; p$, $b = (y^k \, T) \; mod \; p$, где T – исходное сообщение, (a, b) – зашифрованное сообщение.

Дешифрование сообщения выполняется по следующей формуле $T = (b \, (a^x)^{-1}) \; mod \, p$ или $T = (b \, a^{p-1-x}) \; mod \, p$, где $(a^x)^{-1}$ – обратное значение числа $a^x$ по модулю *p*.

Пример шифрования и дешифрования по алгоритму Эль-Гамаля при $k = 7$ приведен в таблице, хотя для шифрования каждого блока (в нашем случае буквы) исходного сообщения надо использовать свое случайное число *k*.

Первая часть шифрованного сообщения – $a = 5^7 \; mod \, 23 = 17$.

$a^x = 17^3 = 4917$, $(a^x)^{-1} = 5$ ($4913 * 5 \; mod \; 23 = 1$) или $a^{p-1-x} = 17^{23-1-3} = 23907243568515132484715 3$.

Таблица 4.2. Пример шифрования по алгоритму Эль-Гамаля (при *k* = const)

| Открытое сообщение, T | Символ | А | Б | Р | А | М | О | В |
|---|---|---|---|---|---|---|---|---|
| | Код | 1 | 2 | 18 | 1 | 14 | 16 | 3 |
| Вторая часть шифрограммы, $b = (32^7 * T) \bmod 37$ | | 19 | 1 | 9 | 19 | 7 | 8 | 20 |
| Открытое сообщение, $T = (b * 2) \bmod 37$ | | 1 | 2 | 18 | 1 | 14 | 16 | 3 |

Ввиду того, что число *k* является произвольным, то такую схему еще называют схемой *вероятностного шифрования*. Вероятностный характер шифрования является преимуществом для схемы Эль-Гамаля, т.к. у схем вероятностного шифрования наблюдается большая стойкость по сравнению со схемами с определенным процессом шифрования. Недостатком схемы шифрования Эль-Гамаля является удвоение длины зашифрованного текста по сравнению с начальным текстом. Для схемы вероятностного шифрования само сообщение $T$ и ключ не определяют шифртекст однозначно. В схеме Эль-Гамаля необходимо использовать различные значения случайной величины $k$ для шифровки различных сообщений $T$ и $T'$. Если использовать одинаковые $k$, то для соответствующих шифртекстов $(a, b)$ и $(a', b')$ выполняется соотношение $b\ (b')^{-1} = T\ (T')^{-1}\ (mod\ p)$. Из этого выражения можно легко вычислить $T$, если известно $T'$.

### Задание

Зашифровать своё имя по алгоритму Эль-Гамаля с помощью следующих значений.

**Варианты:**

| № | p | x | k |
|---|---|---|---|
| 1. | 19 | 5 | 8 |
| 2. | 17 | 8 | 5 |
| 3. | 23 | 13 | 10 |
| 4. | 29 | 10 | 13 |
| 5. | 19 | 14 | 10 |
| 6. | 39 | 10 | 14 |
| 7. | 13 | 11 | 7 |
| 8. | 31 | 7 | 11 |
| 9. | 29 | 6 | 12 |
| 10. | 29 | 12 | 14 |
| 11. | 19 | 14 | 6 |
| 12. | 13 | 12 | 13 |
| 13. | 23 | 13 | 12 |
| 14. | 41 | 11 | 5 |
| 15. | 11 | 5 | 11 |
| 16. | 13 | 6 | 8 |
| 17. | 23 | 8 | 6 |
| 18. | 37 | 13 | 8 |
| 19. | 41 | 8 | 13 |
| 20. | 11 | 6 | 7 |
| 21. | 23 | 7 | 6 |
| 22. | 29 | 9 | 8 |
| 23. | 31 | 8 | 9 |

## Практическая работа № 5

**Тема:** Разработка программного обеспечения, позволяющего слаживать и скалярно умножать точки на эллиптических линиях.

**Цель работы:** Формирование знаний и навыков по разработке программного средства и решению задач по эллиптическим кривым.

### Теоретическая часть

*Эллиптическая криптография* — раздел криптографии, который изучает асимметричные криптосистемы, основанные на эллиптических кривых над конечными полями. Основное преимущество эллиптической криптографии заключается в том, что на сегодняшний день не известно существование субэкспоненциальных алгоритмов решения задачи дискретного логарифмирования.

Эллиптическая кривая над полем $K$ - неособая кубическая кривая на проективной плоскости над $K'$ (алгебраическим замыканием поля $K$ задаваемая уравнением 3-й степени с коэффициентами из поля $K$ и «точкой на бесконечности». В подходящих аффинных координатах её уравнение приводится к виду:

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_5.$$

*Каноническая форма*

Если $char\ K$ (характеристика поля $K$) не равна 2 или 3, то уравнение с помощью замены координат приводится к канонической форме (форме Вейерштрасса):

$$y^2 = x^3 + ax + b.$$

Если char K=3, то каноническим видом уравнения является вид:

$$y^2 = x^3 + a_2x^2 + a_4x + a_6.$$

Если char K=2, то уравнение приводится к одному из видов:

$y^2 + ay = x^3 + bx + c$ — суперсингулярные кривые или

$y^2 + axy = x^3 + bx^2 + c$ — несуперсингулярные кривые.

*Эллиптические кривые над вещественными числами*

Формальное определение эллиптической кривой требует некоторых знаний в алгебраической геометрии, но некоторые свойства эллиптических кривых над вещественными числами можно описать, используя только знания алгебры и геометрии старших классов школы.

Поскольку характеристика поля вещественных чисел — 0, а не 2 или 3, то эллиптическая кривая — плоская кривая, определяемая уравнением вида:

$$y^2 = x^3 + ax + b$$

где а и $b$ — вещественные числа. Этот вид уравнений называется уравнениями Вейерштрасса.

Определение эллиптической кривой также требует, чтобы кривая не имела особых точек. Геометрически это значит, что график не должен иметь каспов и самопересечений. Алгебраически, достаточно проверить, что дискриминант

$$\Delta = -16(4a^3 + 27b^2)$$

не равен нулю.

Если кривая не имеет особых точек, то её график имеет две связные компоненты, если дискриминант положителен, и одну — если отрицателен. Например, для графиков выше в первом случае дискриминант равен 64, а во втором он равен −368.

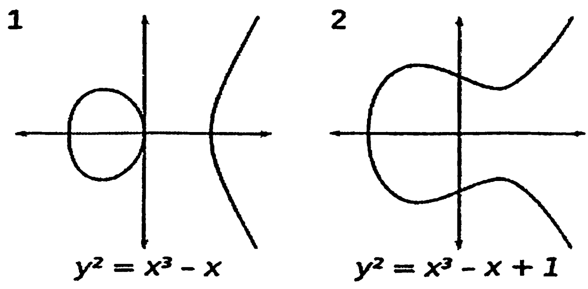



Рис. 5.1. Графики кривых $y^2 = x^3 - x$ и $y^2 = x^3 - x + 1$

*Групповой закон*

Добавлением «точки в бесконечности» получается проективный вариант этой кривой. Если $P$ и $Q$ — две точки на кривой, то возможно единственным образом описать третью точку — точку пересечения данной кривой с прямой, проведённой через $P$ и $Q$. Если прямая является касательной к кривой в точке, то такая точка считается дважды. Если прямая параллельна оси ординат, третьей точкой будет точка в бесконечности.

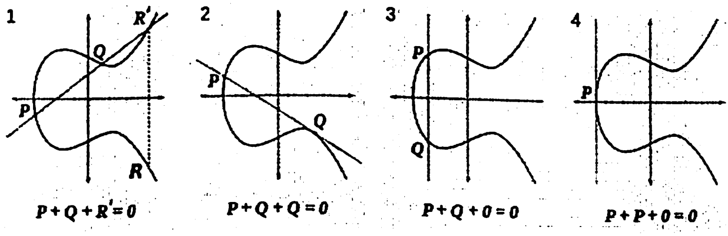



Рис. 5.2. Типы сложения точек в эллиптических кривых

Таким образом, можно ввести групповую операцию «+» на кривой со следующими свойствами: точка в бесконечности (обозначаемая символом $O$) является нейтральным элементом группы, и если прямая пересекает данную кривую в точках $P$, $Q$ и $R'$, то $P+Q+R'=O$ в группе. Суммой точек $P$ и $Q$ называется точка $R=P+Q$, которая симметрична точке $R'$ относительно оси $Ox$. Можно показать, что относительно введённой таким образом операции лежащие на кривой точки и точка $O$ образуют абелеву группу; в частности, свойство ассоциативности операции «+» можно доказать, используя теорему о 9 точках на кубической кривой (кубике).

Данная группа может быть описана и алгебраически. Пусть дана кривая $y^2 = x^3 + ax + b$ над полем $K$ (характеристика которого не равна ни 2, ни 3), и точки $P = (x_P, y_P)$ и $Q = (x_Q, y_Q)$ на кривой; допустим, что $x_P \neq x_Q$. Пусть $s = \frac{y_P - y_Q}{x_P - x_Q}$; так как $K$ — поле, то $s$ строго определено. Тогда мы можем определить $R=P+Q=(x_R, y_R)$ следующим образом:

$$x_R = s^2 - x_P - x_Q,$$

$$y_R = -y_P + s(x_P - x_R).$$

Если $x_Q = x_P$ то есть два варианта. Если $y_P = -y_Q$, то сумма определена как *0*; значит, обратную точку к любой точке на кривой можно найти, отразив её относительно оси *Ox*. Если $y_P = -y_Q \neq 0$, то $R = P + P = 2P = (x_R, y_R)$ определяется так:

$$s = \frac{3x^2{}_P + a}{2y_P} \qquad x_R = s^2 - 2x_P \qquad y_R = -y_P + s(x_P - x_R)$$

Если $y_P = -y_Q = 0$, то $P + P = 0$

Обратный элемент к точке *P*, обозначаемый *-P* и такой, что *P+(-P)=0*, в рассмотренной выше группе определяется так:

Если координата $y_p$, точки $P = (x_P, y_P)$ не равна 0, то $-P = (x_P, -y_P)$.

Если $y_p = 0$, то $-P = P = (x_P, y_P)$.

Если $P = 0$ — точка на бесконечности, то и $-P = 0$.

Точка *Q=nP*, где *n* целое, определяется (при *n*>0) как $Q = \underbrace{P + P \ldots + P}_{n}$. Если *n*<0, то *Q* есть обратный элемент к |*n*| *P*. Если *n*=0, то *Q*=0 *P*=O. Для примера покажем, как найти точку *Q*=4*P*: она представляется как 4*P*=2*P*+2*P*, а точка 2*P* находится по формуле 2*P*=*P*+*P*.

Таблица 5.1 Канонические уравнения с выражениями арифметических операций для приведенных случаев.

| Тип поля и вариант кривой | Каноническое уравнение кривой | Формула сложения: $(x,y)=(x_1,y_1)+(x_2,y_2)$, $x_1 \neq x_2$ | Формула удвоения: $(x,y)=(x_1,y_1)+(x_1,y_1)$, |
|---|---|---|---|
| Поле характеристики, отличной от 2 и 3 | $y^2 = x^3 + ax + b$ | $x = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)^2 - x_1 - x_2$ <br> $y = -y_1 + \frac{y_2 - y_1}{x_2 - x_1}(x_1 - x)$ | $x = \left(\frac{3x_1^2 + a}{2y_1}\right)^2 - 2x_1$ <br> $y = -y_1 + \frac{3x_1^2 + a}{2y_1}(x_1 - x)$ |
| Поле характеристики 3 | $y^2 = x^3 + ax^2 + bx + c$ | $x = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)^2 - x_1 - x_2 - a$ <br> $y = -y_1 + \frac{y_2 - y_1}{x_2 - x_1}(x_1 - x)$ | $x = \left(\frac{ax_1 - b}{y_1}\right)^2 - a + x_1$ <br> $y = -y_1 + \frac{ax_1 - b}{y_1}(x_1 - x)$ |
| Поле характеристики 2, суперсингулярная кривая | $y^2 + ay = x^3 + bx + c$ | $x = \left(\frac{y_2 + y_1}{x_2 + x_1}\right)^2 + x_1 + x_2$ <br> $y = y_1 + a + \frac{y_2 + y_1}{x_2 + x_1}(x_1 + x)$ | $x = \frac{x_1^4 + b^2}{a^2}$ <br> $y = y_1 + a + \frac{x_1^2 + b}{a}(x_1 + x)$ |
| Поле характеристики 2, несуперсингулярная кривая | $y^2 + axy = x^3 + bx^2 + c$ | $x = \left(\frac{y_2 + y_1}{x_2 + x_1}\right)^2 + \frac{y_2 + y_1}{x_2 + x_1} + x_1 + x_2 + b$ <br> $y = y_1 + x + \frac{y_2 + y_1}{x_2 + x_1}(x_1 + x)$ | $x = x_1^2 + \frac{y_1^2}{x_1^2} + x_1 + \frac{y_1}{x_1} + b$ <br> $y = x_1^2 + \frac{x_1^2 + y_1}{x_1}x + x$ |

### Задание

Вычислите сложения и скалярные умножения точек.

$F(x) = E_{23}(1,0)$

| 1. | P | Q | N*P=? | P+Q=? |
|---|---|---|---|---|
| 2. | (0,0) | (1,18) | 23P | |
| 3. | (1,5) | (9,5) | 23P | |
| 4. | (9,5) | (9,18) | 23P | |
| 5. | (9,18) | (11.10) | 23P | |
| 6. | (11,13) | (11,15) | 23P | |
| 7. | (11,15) | (13,5) | 23P | |
| 8. | (13,18) | (15,3) | 23P | |
| 9. | (15,3) | (15,20) | 23P | |
| 10. | (15,20) | (16,8) | 23P | |
| 11. | (16,8) | (16,15) | 23P | |
| 12. | (16,15) | (17,10) | 23P | |
| 13. | (17,10) | (17,13) | 23P | |
| 14. | (17,13) | (18,10) | 23P | |
| 15. | (18,10) | (18,13) | 23P | |
| 16. | (18,13) | (19,1) | 23P | |
| 17. | (19,1) | (19,22) | 23P | |
| 18. | (19,22) | (20,4) | 23P | |
| 19. | (20,4) | (20,19) | 23P | |
| 20. | (20,19) | (21,6) | 23P | |
| 21. | (21,6) | (21,17) | 23P | |
| 22. | (21,17) | (20,4) | 23P | |
| 23. | (20,4) | (17,13) | 23P | |
| 24. | (17,13) | (16,8) | 23P | |
| 25. | (11,13) | (9,5) | 23P | |

## Практическая работа № 6

**Тема:** Разработка программного средства алгоритма шифрования с открытым ключом Рабина

**Цель работы:** Формирование знаний и навыков по разработке программного средства и решению задач по алгоритму Рабина.

### Теоретическая часть

Криптосистема Рабина — криптографическая система с открытым ключом, безопасность которой обеспечивается сложностью поиска квадратных корней в кольце остатков по модулю составного числа. Безопасность системы, как и безопасность метода RSA, обусловлена сложностью разложения на множители больших чисел. Зашифрованное сообщение можно расшифровать 4 способами. Недостатком системы является необходимость выбора истинного сообщения из 4-х возможных.

*Генерация ключа*

Система Рабина, как и любая асимметричная криптосистема, использует открытый и закрытый ключи. Открытый ключ используется для шифрования сообщений и может быть опубликован для всеобщего обозрения. Закрытый ключ необходим для расшифровки и должен быть известен только получателям зашифрованных сообщений.

Процесс генерации ключей следующий:

- выбираются два случайных числа $p$ и $q$ с учётом следующих требований:
  - числа должны быть большими (см. разрядность);
  - числа должны быть простыми;
  - должно выполняться условие: $p \equiv q \equiv 3 \bmod 4$.

  Выполнение этих требований сильно ускоряет процедуру извлечения корней по модулю $p$ и $q$;
- вычисляется число $n = p \cdot q$;
- число n — открытый ключ; числа p и q — закрытый.

Пример. Пусть $p = 7$ и $q = 11$. Тогда $n = p * q = 7 * 11 = 77$. Число n = 77 — открытый ключ, а числа $p = 7$ и $q = 11$ — закрытый. Получатель сообщает отправителям число 77. Отправители шифруют сообщение, используя число 77, и отправляют получателю. Получатель расшифровывает сообщение с помощью чисел 7 и 11. Приведённые ключи плохи для практического использования, так как число 77 легко раскладывается на простые множители (7 и 11).

**Практическая часть**

*Шифрование*

Исходное сообщение m (текст) шифруется с помощью открытого ключа — числа n по следующей формуле:

$c = m^2 \bmod n$.

Благодаря использованию умножения по модулю скорость шифрования системы Рабина больше, чем скорость шифрования по методу RSA, даже если в последнем случае выбрать небольшое значение экспоненты.

Пример (продолжение). Пусть исходным текстом является $m = 20$. Тогда зашифрованным текстом будет: $c = m^2 \bmod n = 20^2 \bmod 77 = 400 \bmod 77 = 15$.

*Расшифрование*

Для расшифровки сообщения необходим закрытый ключ — числа $p$ и $q$. Процесс расшифровки выглядит следующим образом:

сначала, используя алгоритм Евклида, из уравнения $y_p * p + y_q * q = 1$ находят числа $y_p$ и $y_q$;

далее, используя китайскую теорему об остатках, вычисляют четыре числа:

$r = y_p * p * m_q + y_q * q * m_p \ (mod\, n)$

$-r = n - r$

$s = y_p * p * m_q - y_q * q * m_p \ (mod\, n)$

$s = n - s$

Одно из этих чисел является истинным открытым текстом $m$. Пример (окончание). В результате расшифровки получаем: $m \in \{64, 20, 13, 57\}$. Видим, что один из корней является исходным текстом $m$.

## Задание

Зашифруйте свою фамилию и имя с помощью алгоритма Рабина

## Практическая работа № 7

**Тема:** Шифрование данных гибридным методом шифрования с использованием библиотеки OpenSSL

**Цель работы:** Формирование знаний и навыков по шифрованию гибридным методом с помощью библиотеки OpenSSL.

### Теоретическая часть

*Гибридная (или комбинированная) криптосистема* — это система шифрования, совмещающая преимущества криптосистемы с открытым ключом с производительностью симметричных криптосистем. Симметричный ключ используется для шифрования данных, а асимметричный для шифрования самого симметричного ключа, иначе это называется числовой упаковкой.

*Принцип*

Криптографические системы используют преимущества двух основных криптосистем: симметричной и асимметричной криптографии. На этом принципе построены такие протоколы, как PGP и TLS.

Основной недостаток асимметричной криптографии состоит в низкой скорости из-за сложных вычислений, требуемых её алгоритмами, в то время как симметричная криптография традиционно показывает высокую скорость работы. Однако симметричные криптосистемы имеет один существенный недостаток — её использование предполагает наличие защищённого канала для передачи ключей. Для преодоления этого недостатка прибегают к асимметричным криптосистемам, которые используют пару ключей: открытый и закрытый.

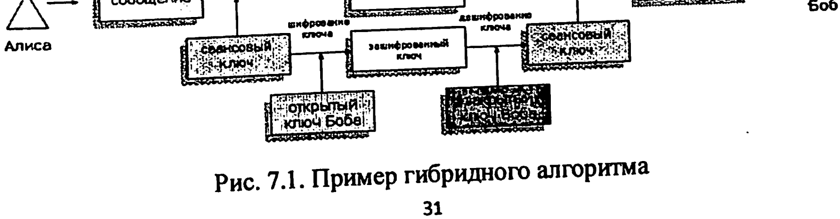



Рис. 7.1. Пример гибридного алгоритма

*Этап отправки:*

- Алиса генерирует случайный сеансовый ключ
- сообщение Алисы шифруется сеансовым ключом (с помощью симметричного алгоритма)
- сеансовый ключ шифруется открытым ключом Боба (асимметричным алгоритмом)
- Алиса посылает Бобу зашифрованное сообщение и зашифрованный сеансовый ключ

*Этап приёма:*

- Боб получает зашифрованное сообщение Алисы и зашифрованный сеансовый ключ
- Боб расшифровывает сеансовый ключ своим закрытым ключом
- при помощи полученного, таким образом, сеансового ключа Боб расшифровывает зашифрованное сообщение Алисы

*Шифрование*

Большинство гибридных систем работают следующим образом. Для симметричного алгоритма (3DES, IDEA, AES или любого другого) генерируется случайный сеансовый ключ. Такой ключ как правило имеет размер от 128 до 512 бит (в зависимости от алгоритма). Затем используется симметричный алгоритм для шифрования сообщения. В случае блочного шифрования необходимо использовать режим шифрования (например CBC), что позволит шифровать сообщение с длиной, превышающей длину блока. Что касается самого случайного ключа, он должен быть зашифрован с помощью открытого ключа получателя сообщения, и именно на этом этапе применяется криптосистема с открытым ключом (RSA или Алгоритм Диффи — Хеллмана). Поскольку сеансовый ключ короткий, его шифрование занимает немного времени. Шифрование набора сообщений с помощью асимметричного алгоритма — это задача вычислительно более сложная, поэтому здесь предпочтительнее использовать симметричное шифрование.

Затем достаточно отправить сообщение, зашифрованное симметричным алгоритмом, а также соответствующий ключ в зашифрованном виде. Получатель сначала расшифровывает ключ с помощью своего секретного ключа, а затем с помощью полученного ключа получает и всё сообщение.

## Практическая часть

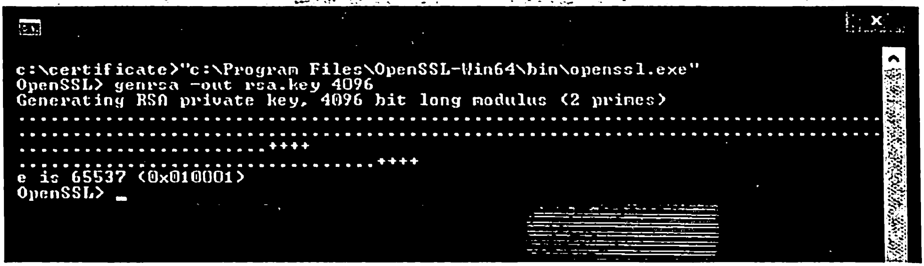



Рис.7.2. Генерация личного ключа

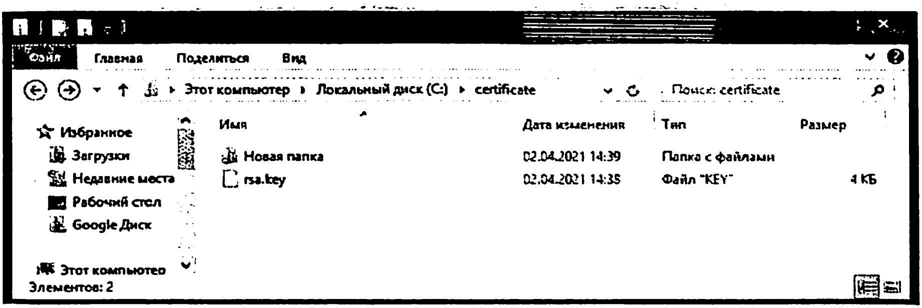



Рис.7.3. Сгенерированный личный ключ в виде файла

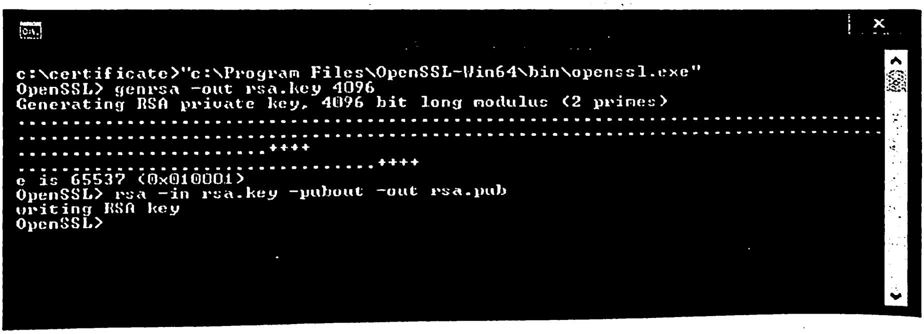



Рис.7.4. Генерация открытого ключа

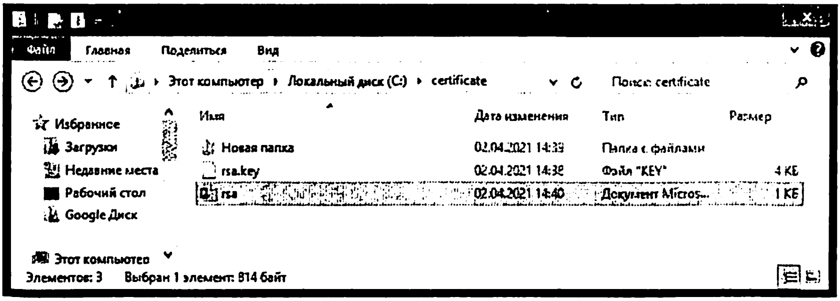



Рис.7.5. Сгенерированный открытый ключ в виде файла

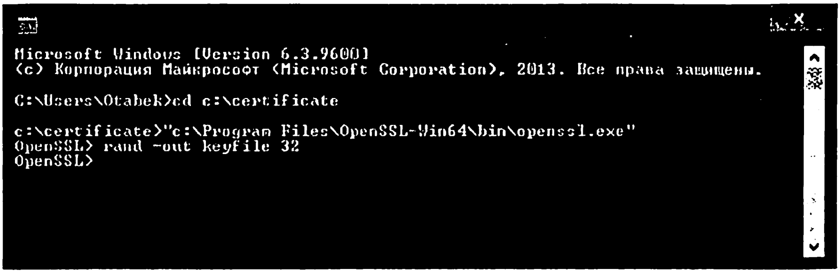



Рис.7.6. Генерация ключа для симметричного шифрования

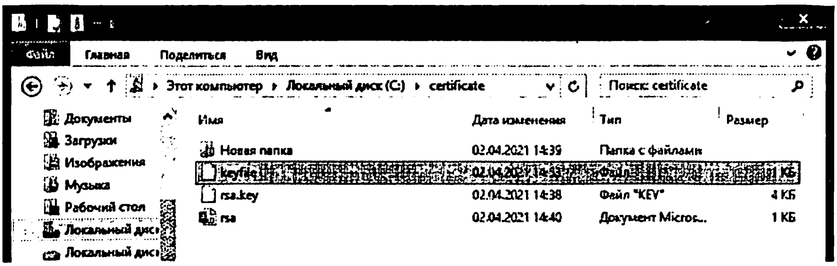



Рис.7.7. Сгенерированный ключ keyfile

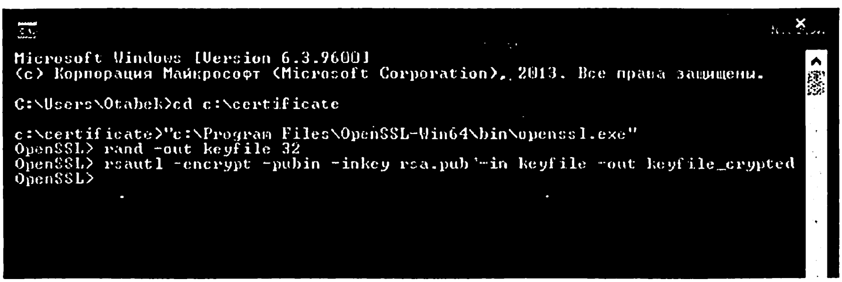



Рис.7.8. Шифрование ключа для симметричного шифрования

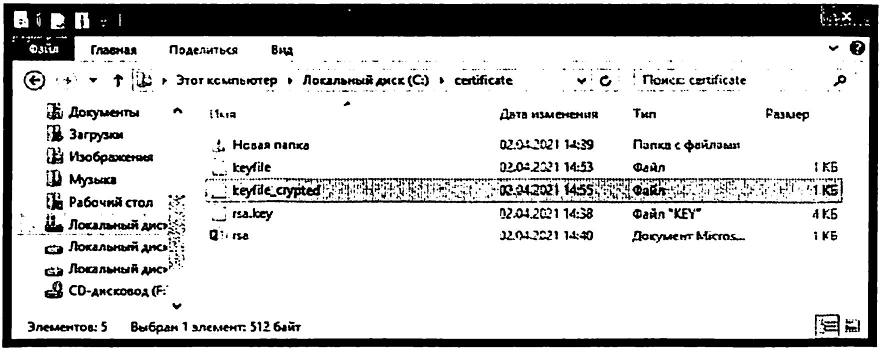



Рис.7.9. Зашифрованный симметричный ключ keyfile_crypted

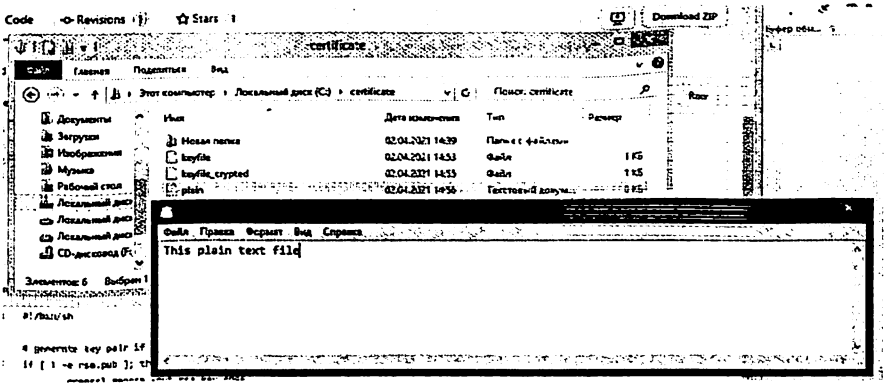



Рис.7.10. Содержание файла открытого текста

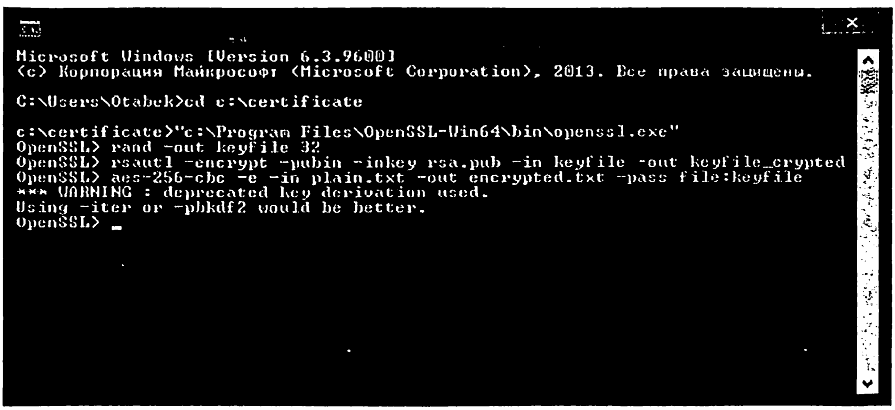



Рис.7.11. Шифрование открытого текста

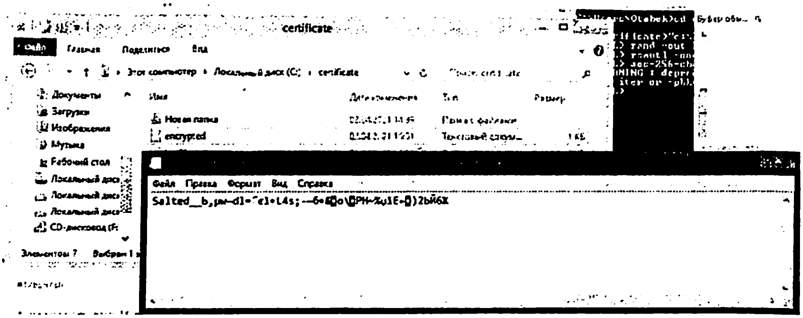



Рис.7.12. Содержание файла зашифрованного текста

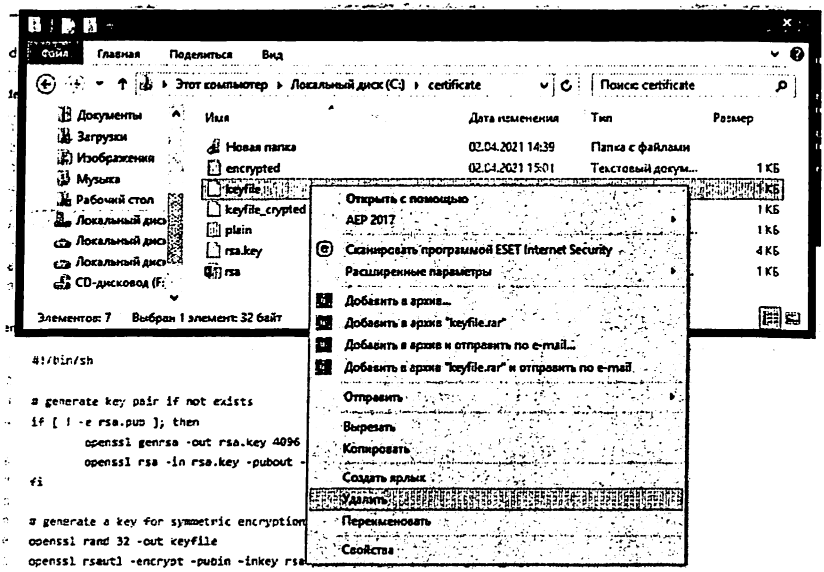



Рис.7.13. Удаление симметричного ключа

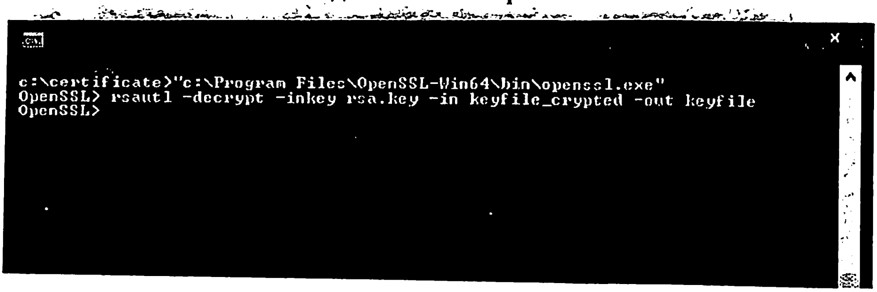



Рис.7.14. Расшифрование зашифрованного симметричного ключа

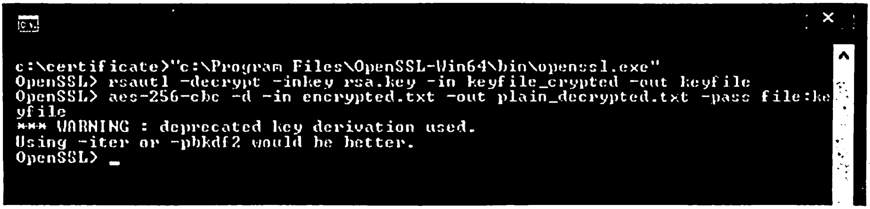



Рис.7.15. Использование симметричного ключа для расшифровки шифротекста

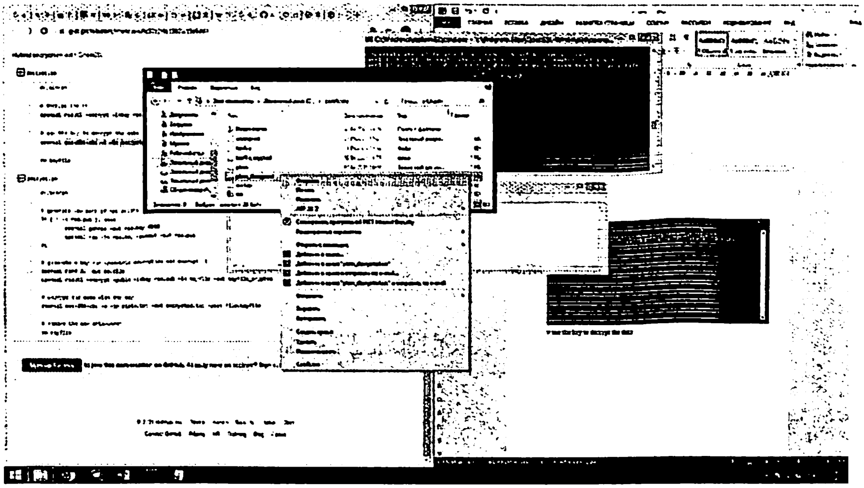


Рис.7.16. Открытие файла расшифрованного теста

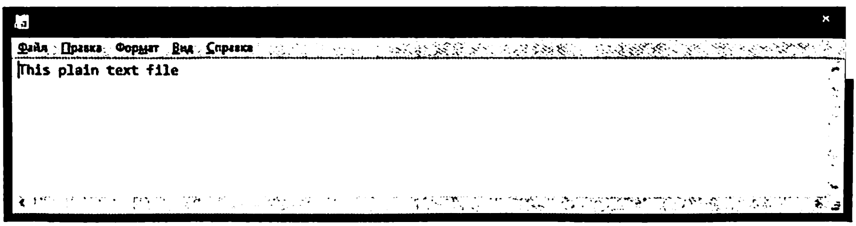



Рис.7.17. Открытый текст после расшифровки

## Задание

Зашифруйте и расшифруйте файл используя гибридное шифрование. В файле написать свою фамилию и имя.

## Практическая работа № 8

**Тема:** Создания ЭЦП на основе алгоритма RSA в библиотеке OpenSSL.

**Цель работы: Формирование знаний и навыков по создания ЭЦП на основе алгоритма RSA с помощью библиотеки OpenSSL.**

### Теоретическая часть

*Цифровая подпись*

Система RSA может использоваться не только для шифрования, но и для цифровой подписи.

Предположим, что Алисе (стороне $A$) нужно отправить Бобу (стороне $B$) сообщение $m$, подтверждённое электронной цифровой подписью.


Рис 8.1. Схема ЭЦП

*Отправитель:*

- Взять открытый текст $m$
- Создать цифровую подпись $s$ с помощью своего секретного ключа $\{d,n\}$: $s = S_A(m) = m^d \ mod \ n$
- Передать пару $\{m,s\}$состоящую из сообщения и подписи.

*Получатель:*

- Принять пару $\{m,s\}$
- Взять открытый ключ $\{e,n\}$ Алисы
- Вычислить прообраз сообщения из подписи:
- Проверить подлинность подписи (и неизменность сообщения), сравнив $m$ и $m'$

*Действия, выполняемые отправителем*

Чтобы сгенерировать закрытый (и открытый ключ):

```
$ openssl genpkey -algorithm RSA -pkeyopt rsa_keygen_bits:2048 -
pkeyopt rsa_keygen_pubexp:3 -out privkey-ID.pem
$ cat privkey-ID.pem
-----BEGIN PRIVATE KEY-----
MIIEvQIBADANBgkqhkiG9w0BAQEFAASCBKcwggSjAgEAAoIBAQCoXEAmbAuh9Nks
xtjIqgW8+MjaoRLWIKOpr54E7XcpzMSlNZggPBp0sLjfgvNFBPP7BrQms3qigwow
krML/fdwSFybigmuTCyJS/UIn3J5s70vUSpQ9M8oAU+6lvRdiByqR0zBnnWdR9B8
wW2/jM2Ng3yq51S6qR6LUs92jEzYATzldf8z+qcUL+navmOSLdAll0qQpbKjEjIl
esJIkqrKlQiulN0TQbexC9dNwtI79G79UR+YOR8CWJyYy/ZPeUrsrlmcSGL7facW
/aG2hh85/XdICm2PWgRySUu0M2rHdxL+AMukauYnlw4gddTO0cmUNyxKrVr5aQBP
hZxKtFV5AgEDAoIBAHA9gBmdXRajO3MvOzBxWSil2zxrYeQVwnEfvq3zpMaIgxjO
ZWrSvE3LJepXTNit9/yvIsR3pxcCBssMdllT+kra6GexW8mIHbDdTgW/oaZ303Tg
xuCjNMVWNScPTZOwExwviIEUTmjaiv3WSSpd3l5XqHHvjdHGFFzh36RdiI//vcSX
VHC76AkhkJl3aDEIUSQPMfE0OmI4dgK2sxH8BXAmAgc7YOksLF4t+tjaEoeUFQWP
SwFiGgVaU3wtmvlDoSwbAKSWs/9hDg3vgN8AFku3HCdBkpmpp2CYqoBWFDfUNW2q
TtB7IU2fwUOtoqiW8CegqVNf+X+KWT85mblNnqMCgYEA3z2IhWyENYsHRrfbpISR
q3y5l5sgFMlofRbPA5AZbZANY48jFPSeuKWJ1HhhZpwai+dcKf5R2w5V/4vpKqec
wFFGkXiOshkzty/67A75Uww/iewff0nj8ZwG7oLYl2PHu7iyyHiwbTj7N2lRapq+
iUHpd4RBpiOPoad4lD+CDWcCgYEAwREKex5clXt2SjavosQPqwMG6Au3RkJVBBqZ
shl/NRJOohTYtsDgvH49CpAaT9R7w42eBRfUHOv7H9KeYyv3GNlARyzXouM4WtIb
dFkMqrwrQyEIkl73l8VdXXDZtQ/xByDOjPMBxvosNM2f9jcw2Bbctslbvpa J2Mk2
oW892h8CgYEAlNOwWPMCzlyvhHqSba22clMmZRIVYzOa/g80rQq7nmAI7QoXY02/
JcOxOFBA7xK8XUToG/7hPLQ5VQfwxxpogDYvC6W0drt3z3VR8rSmNll/sUgU/4aX
9mgEnwHlukKFJ9B3MFBlniX8z542RxHUW4FGT62BGW0Ka8T7DX+sCO8CgYEAgLYG
/L7oY6ekMXnKbIKlHKyvRV0k2YGOArxmdr5UzgwObA3lzytAfal+Bwq8NThSgl5p
WLqNaJlSFTcUQhlPZeYq2h3lF0IlkeFnouYIcdLHghYFtun6ZS4+Pks7zgqgr2s0
XfdWhKbIIzO/+XogkA89zzDnlGRb5dt5wPTT5r8CgYEA29235n/Hw7wzOJyao6nO
3rjCZon4/V2G800VJF5hhAqCX5KDLd0KIMbaHaxsjW+n79CqZSUz3kZtpSXBXRJ7
SIXoCYljaoxdJ6SkVED6uFmcZ+3iwioxXzpIFIW0ZZj5S/WgBkPsioAJ6Cp5S8zh
BFB15UA+JWFH2SRabjXf0+4=
-----END PRIVATE KEY-----
```

Закрытый ключ закодирован с помощью Base64. Чтобы просмотреть значения:

```
$ openssl pkey -in privkey-ID.pem -text
-----BEGIN PRIVATE KEY-----
MIIEvQIBADANBgkqhkiG9w0BAQEFAASCBKcwggSjAgEAAoIBAQCoXEAmbAuh9Nks
xtjIqgW8+MjaoRLWIKOpr54E7XcpzMSlNZggPBp0sLjfgvNFBPP7BrQms3qigwow
krML/fdwSFybigmuTCyJS/UIn3J5s70vUSpQ9M8oAU+6lvRdiByqR0zBnnWdR9B8
wW2/jM2Ng3yq51S6qR6LUs92jEzYATzldf8z+qcUL+navmOSLdAll0qQpbKjEjIl
esJIkqrKlQiulN0TQbexC9dNwtI79G79UR+YOR8CWJyYy/ZPeUrsrlmcSGL7facW
/aG2hh85/XdICm2PWgRySUu0M2rHdxL+AMukauYnlw4gddTO0cmUNyxKrVr5aQBP
hZxKtFV5AgEDAoIBAHA9gBmdXRajO3MvOzBxWSil2zxrYeQVwnEfvq3zpMaIgxjO
ZWrSvE3LJepXTNit9/yvIsR3pxcCBssMdllT+kra6GexW8mIHbDdTgW/oaZ303Tg
xuCjNMVWNScPTZOwExwviIEUTmjaiv3WSSpd3l5XqHHvjdHGFFzh36RdiI//vcSX
VHC76AkhkJl3aDEIUSQPMfE0OmI4dgK2sxH8BXAmAgc7YOksLF4t+tjaEoeUFQWP
SwFiGgVaU3wtmvlDoSwbAKSWs/9hDg3vgN8AFku3HCdBkpmpp2CYqoBWFDfUNW2q
TtB7IU2fwUOtoqiW8CegqVNf+X+KWT85mblNnqMCgYEA3z2IhWyENYsHRrfbpISR
q3y5l5sgFMlofRbPA5AZbZANY48jFPSeuKWJ1HhhZpwai+dcKf5R2w5V/4vpKqec
wFFGkXiOshkzty/67A75Uww/iewff0nj8ZwG7oLYl2PHu7iyyHiwbTj7N2lRapq+
iUHpd4RBpiOPoad4lD+CDWcCgYEAwREKex5clXt2SjavosQPqwMG6Au3RkJVBBqZ
shl/NRJOohTYtsDgvH49CpAaT9R7w42eBRfUHOv7H9KeYyv3GNlARyzXouM4WtIb
```

```
dFkMqrwrQyEIkl73l8VdXXDZtQ/xByDOjPMBxvosNM2f9jcw2BbctslbvpaJ2Mk2
oW892h8CgYEAlNOwWPMCzlyvhHqSba22clMmZRIVYzOa/g80rQq7nmAI7QoXY02/
JcOxOFBA7xK8XUToG/7hPLQ5VQfwxxpogDYvC6W0drt3z3VR8rSmNll/sUgU/4aX
9mgEnwHlukKFJ9B3MFBlniX8z542RxHUW4FGT62BGW0Ka8T7DX+sCO8CgYEAgLYG
/L7oY6ekMXnKbIKlHKyvRV0k2YGOArxmdr5Uzgw0bA3lzytAfal+Bwq8NThSgl5p
WLqNaJlSFTcUQhlPZeYq2h3lF0IlkeFnouYIcdLHghYFtun6ZS4+Pks7zgqgr2s0
XfdWhKbIIzO/+XogkA89zzDnlGRb5dt5wPTT5r8CgYEA29235n/Hw7wzOJyao6nO
3rjCZon4/V2G800VJF5hhAqCX5KDLd0KIMbaHaxsjW+n79CqZSUz3kZtpSXBXRJ7
SIXoCYljaoxdJ6SkVED6uFmcZ+3iwioxXzpIFIW0ZZj5S/WgBkPsioAJ6Cp5S8zh
BFBl5UA+JWFH2SRabjXf0+4=
```


```
-----END PRIVATE KEY-----
Private-Key: (2048 bit)
modulus:
    00:a8:5c:40:26:6c:0b:a1:f4:d9:2c:c6:d8:c8:aa:
    05:bc:f8:c8:da:a1:12:d6:20:a3:a9:af:9e:04:ed:
    77:29:cc:c4:a5:35:98:20:3c:1a:74:b0:b8:df:82:
    f3:45:04:f3:fb:06:b4:26:b3:7a:a2:83:0a:30:92:
    b3:0b:fd:f7:70:48:5c:9b:8a:09:ae:4c:2c:89:4b:
    f5:08:9f:72:79:b3:bd:2f:51:2a:50:f4:cf:28:01:
    4f:ba:96:f4:5d:88:1c:aa:47:4c:c1:9e:75:9d:47:
    d0:7c:c1:6d:bf:8c:cd:8d:83:7c:aa:e7:54:ba:a9:
    1e:8b:52:cf:76:8c:4c:d8:01:3c:f5:75:ff:33:fa:
    a7:14:2f:e9:da:be:63:92:2d:d0:35:d7:4a:90:a5:
    b2:a3:12:32:35:7a:c2:48:92:aa:ca:95:08:ae:d4:
    dd:13:41:b7:b1:0b:d7:4d:c2:d2:3b:f4:6e:fd:51:
    1f:98:39:1f:02:58:9c:98:cb:f6:4f:79:4a:ec:af:
    59:9c:48:62:fb:7d:a7:16:fd:a1:b6:86:1f:39:fd:
    77:48:0a:6d:8f:5a:04:72:49:4b:b4:33:6a:c7:77:
    12:fe:00:cb:a4:6a:e6:27:97:0e:20:75:d4:ce:d1:
    c9:94:37:2c:4a:ad:5a:f9:69:00:4f:85:9c:4a:b4:
    55:79
publicExponent: 3 (0x3)
privateExponent:
    70:3d:80:19:9d:5d:16:a3:3b:73:2f:3b:30:71:59:
    28:a5:db:3c:6b:61:e4:15:c2:71:1f:be:ad:f3:a4:
    c6:88:83:18:ce:65:6a:d2:bc:4d:cb:25:ea:57:4c:
    d8:ad:f7:fc:af:22:c4:77:a7:17:02:06:cb:0c:77:
    5d:53:fa:4a:da:e8:67:b1:5b:c9:88:1d:b0:dd:4e:
    05:bf:a1:a6:77:d3:74:e0:c6:e0:a3:34:c5:56:35:
    27:0f:4d:93:b0:13:1c:2f:88:81:14:4e:68:da:8a:
    fd:d6:49:2a:5d:de:5e:57:a8:71:ef:8d:d1:c6:14:
    5c:e1:df:a4:5d:88:8f:ff:bd:c4:97:54:70:bb:e8:
    09:21:90:9d:77:68:31:08:51:24:0f:31:f1:34:3a:
    62:38:76:02:b6:b3:11:fc:05:70:26:02:07:3b:60:
    e9:2c:2c:5e:2d:fa:d8:da:12:87:94:15:05:8f:4b:
    01:62:1a:05:5a:53:7c:2d:9a:fd:43:a1:2c:1b:00:
    a4:96:b3:ff:61:0e:0d:ef:80:df:00:16:4b:b7:1c:
    27:41:92:99:a9:a7:60:98:aa:80:56:14:37:d4:35:
    6d:aa:4e:d0:7b:21:4d:9f:c1:43:ad:a2:a8:96:f0:
    27:a0:a9:53:5f:f9:7f:8a:59:3f:39:99:bd:4d:9e:
    a3
prime1:
    00:df:3d:88:85:6c:84:35:8b:07:46:b7:db:a4:84:
    91:ab:7c:b9:97:9b:20:14:cd:68:7d:16:cf:03:90:
    19:6d:90:0d:63:8f:23:14:f4:9e:b8:a5:89:d4:78:
    61:66:9c:1a:8b:e7:5c:29:fe:51:db:0e:55:ff:8b:
    e9:2a:a7:9c:c0:51:46:91:78:8e:b2:19:33:b7:2f:
    fa:ec:0e:f9:53:0c:3f:89:ec:1f:7f:49:e3:f1:9c:
```

```
    06:ee:82:d8:97:63:c7:bb:b8:b2:c8:78:b0:6d:38:
    fb:37:6d:51:6a:9a:be:89:41:e9:77:84:41:a6:23:
    8f:a1:a7:78:94:3f:82:0d:67
prime2:
    00:c1:11:0a:7b:1e:5c:95:7b:76:4a:36:af:a2:c4:
    0f:ab:03:06:e8:0b:b7:46:42:55:04:1a:99:b2:1d:
    7f:35:12:4e:a2:14:d8:b6:c0:e0:bc:7e:3d:0a:90:
    1a:4f:d4:7b:c3:8d:9e:05:17:d4:1c:eb:fb:1f:d2:
    9e:63:2b:f7:18:d9:40:47:2c:d7:a2:e3:38:5a:d2:
    1b:74:59:0c:aa:bc:2b:43:21:08:92:5e:f7:97:c5:
    5d:5d:70:d9:b5:0f:f1:07:20:ce:8c:f3:01:c6:fa:
    2c:34:cd:9f:f6:37:30:d8:16:dc:b6:c9:5b:be:96:
    89:d8:c9:36:a1:6f:3d:da:1f
exponent1:
    00:94:d3:b0:58:f3:02:ce:5c:af:84:7a:92:6d:ad:
    b6:72:53:26:65:12:15:63:33:9a:fe:0f:34:ad:0a:
    bb:9e:60:08:ed:0a:17:63:4d:bf:25:c3:b1:38:50:
    40:ef:12:bc:5d:44:e8:1b:fe:e1:3c:b4:39:55:07:
    f0:c7:1a:68:80:36:2f:0b:a5:b4:76:bb:77:cf:75:
    51:f2:b4:a6:37:5d:7f:b1:48:14:ff:86:97:f6:68:
    04:9f:01:e5:ba:42:85:27:d0:77:30:50:75:9e:25:
    fc:cf:9e:36:47:11:d4:5b:81:46:4f:ad:81:19:6d:
    0a:6b:c4:fb:0d:7f:ac:08:ef
exponent2:
    00:80:b6:06:fc:be:e8:63:a7:a4:31:79:ca:6c:82:
    b5:1c:ac:af:45:5d:24:d9:81:8e:02:bc:66:76:be:
    54:ce:0c:34:6c:0d:e5:cf:2b:40:7d:a9:7e:07:0a:
    bc:35:38:52:82:5e:69:58:ba:8d:68:9d:52:15:37:
    14:42:1d:4f:65:e6:2a:da:1d:e5:17:42:25:91:e1:
    67:a2:e6:08:71:d2:c7:82:16:05:b6:e9:fa:65:2e:
    3e:3e:4b:3b:ce:0a:a0:af:6b:34:5d:f7:56:84:a6:
    c8:23:33:bf:f9:7a:20:90:0f:3d:cf:30:e7:d4:64:
    5b:e5:db:79:c0:f4:d3:e6:bf
coefficient:
    00:db:dd:b7:e6:7f:c7:c3:bc:33:38:9c:9a:a3:a9:
    ce:de:b8:c2:66:89:f8:fd:5d:86:f3:4d:15:24:5e:
    61:84:0a:82:5f:92:83:2d:dd:0a:20:c6:da:1d:ac:
    6c:8d:6f:a7:ef:d0:aa:65:25:33:de:46:6d:a5:25:
    c1:5d:12:7b:48:85:e8:09:89:63:6a:8c:5d:27:a4:
    a4:54:40:fa:b8:59:9c:67:ed:e2:c2:2a:31:5f:3a:
    48:14:85:b4:65:98:f9:4b:f5:a0:06:43:ec:8a:80:
    09:e8:2a:79:4b:cc:e1:04:50:75:e5:40:3e:25:61:
```

Чтобы вывести в файл только открытый ключ:

```
$ openssl pkey -in privkey-ID.pem -out pubkey-ID.pem -pubout
$ cat pubkey-ID.pem
-----BEGIN PUBLIC KEY-----
MIIBIDANBgkqhkiG9w0BAQEFAAOCAQ0AMIIBCAKCAQEAqFxAJmwLofTZLMbYyKoF
vPjI2qES1iCjqa+eBO13KczEpTWYIDwadLC434LzRQTz+wa0JrN6ooMKMJKzC/33
cEhcm4oJrkwsiUv1CJ9yebO9L1EqUPTPKAFPupb0XYgcqkdMwZ51nUfQfMFtv4zN
jYN8qudUuqkei1LPdoxM2AE89XX/M/qnFC/p2r5jki3QNddKkKWyoxIyNXrCSJKq
ypUIrtTdE0G3sQvXTcLSO/Ru/VEfmDkfAlicmMv2T31K7K9ZnEhi+32nFv2htoYf
Of13SAptjloEck1LtDNqx3cS/gDLpGrmJ5cOIHXUztHJlDcsSq1a+WkAT4WcSrRV
eQIBAw==
-----END PUBLIC KEY-----
```

Проверьте, посмотрев на исходные значения:

```
$ openssl pkey -in pubkey-ID.pem -pubin -text
-----BEGIN PUBLIC KEY-----
MIIBIDANBgkqhkiG9w0BAQEFAAOCAQ0AMIIBCAKCAQEAqFxAJmwLofTZLMbYyKoF
vPjI2qESliCjqa+eBO13KczEpTWYIDwadLC434LzRQTz+wa0JrN6ooMKMJKzC/33
cEhcm4oJrkwsiUvlCJ9yebO9LlEqUPTPKAFPupb0XYgcqkdMwZ5lnUfQfMFtv4zN
jYN8qudUuqkeilLPdoxM2AE89XX/M/qnFC/p2r5jki3QNddKkKWyoxIyNXrCSJKq
ypUIrtTdE0G3sQvXTcLSO/Ru/VEfmDkfAlicmMv2T31K7K9ZnEhi+32nFv2htoYf
Of13SAptjloEcklLtDNqx3cS/gDLpGrmJ5cOIHXUztHJ1DcsSq1a+WkAT4WcSrRV
eQIBAw==
-----END PUBLIC KEY-----
Public-Key: (2048 bit)
Modulus:
    00:a8:5c:40:26:6c:0b:a1:f4:d9:2c:c6:d8:c8:aa:
    05:bc:f8:c8:da:a1:12:d6:20:a3:a9:af:9e:04:ed:
    77:29:cc:c4:a5:35:98:20:3c:1a:74:b0:b8:df:82:
    f3:45:04:f3:fb:06:b4:26:b3:7a:a2:83:0a:30:92:
    b3:0b:fd:f7:70:48:5c:9b:8a:09:ae:4c:2c:89:4b:
    f5:08:9f:72:79:b3:bd:2f:51:2a:50:f4:cf:28:01:
    4f:ba:96:f4:5d:88:1c:aa:47:4c:c1:9e:75:9d:47:
    d0:7c:c1:6d:bf:8c:cd:8d:83:7c:aa:e7:54:ba:a9:
    1e:8b:52:cf:76:8c:4c:d8:01:3c:f5:75:ff:33:fa:
    a7:14:2f:e9:da:be:63:92:2d:d0:35:d7:4a:90:a5:
    b2:a3:12:32:35:7a:c2:48:92:aa:ca:95:08:ae:d4:
    dd:13:41:b7:b1:0b:d7:4d:c2:d2:3b:f4:6e:fd:51:
    1f:98:39:1f:02:58:9c:98:cb:f6:4f:79:4a:ec:af:
    59:9c:48:62:fb:7d:a7:16:fd:a1:b6:86:1f:39:fd:
    77:48:0a:6d:8f:5a:04:72:49:4b:b4:33:6a:c7:77:
    12:fe:00:cb:a4:6a:e6:27:97:0e:20:75:d4:ce:d1:
    c9:94:37:2c:4a:ad:5a:f9:69:00:4f:85:9c:4a:b4:
    55:79
Exponent: 3 (0x3)
```

Создайте текстовый файл:

```
$ cat message-ID.txt
This is my example message.
```

Чтобы подписать сообщение, вам нужно вычислить его хэш, а затем зашифровать этот хеш своим закрытым ключом. Чтобы создать хеш сообщения (без шифрования):

```
$ openssl dgst -sha1 message-ID.txt
SHA1(message-ID.txt)= 064774b2fb550d8c1d7d39fa5ac5685e2f8b1ca6
```

OpenSSL имеет возможность вычислить хэш и затем подписать его:

```
$ openssl dgst -sha1 -sign privkey-ID.pem -out sign-ID.bin message-
ID.txt
$ ls -l
total 16
-rw-r--r-- 1 sgordon users   28 2012-03-04 15:14 message-ID.txt
-rw-r--r-- 1 sgordon users 1704 2012-03-04 14:58 privkey-ID.pem
-rw-r--r-- 1 sgordon users  451 2012-03-04 15:08 pubkey-ID.pem
```

```
-rw-r--r-- 1 sgordon users  256 2012-03-04 15:20 sign-ID.bin
```

Чтобы зашифровать сообщение с помощью RSA, используйте открытый ключ получателя:

```
$ openssl pkeyutl -encrypt -in message.txt -pubin -inkey pubkey-Steve.pem -out ciphertext-ID.bin
```

Обратите внимание, что прямое шифрование RSA следует использовать только для небольших файлов, длина которых меньше длины ключа. Если вы хотите зашифровать большие файлы, используйте шифрование с симметричным ключом. Два подхода к этому с помощью OpenSSL: (1) сгенерировать случайный ключ, который будет использоваться с симметричным шифром для шифрования сообщения, а затем зашифровать ключ с помощью RSA; (2) используйте операцию `smime` , которая объединяет RSA и симметричный шифр для автоматизации подхода 1.

*Шаги, выполняемые получателем*

Открытый ключ был сгенерирован и предоставлен отправителю:

```
$ openssl genpkey -algorithm RSA -pkeyopt rsa_keygen_bits:2048 -pkeyopt rsa_keygen_pubexp:3 -out privkey-ID.pem
$ openssl pkey -in privkey-Steve.pem -out pubkey-Steve.pem -pubout
```

Чтобы расшифровать полученный зашифрованный текст:

```
$openssl pkeyutl -decrypt -in ciphertext-ID.bin -inkey privkey-Steve.pem -out received-ID.txt
$ cat received-ID.txt
This is my example message.
```

Чтобы проверить подпись сообщения:

```
$ openssl dgst -sha1 -verify pubkey-ID.pem -signature sign-ID.bin received-ID.txt
Verified OK
```

**Задание**

Создание ЭЦП на основе алгоритма RSA с использованием библиотеки OpenSSL.

## Практическая работа №9

**Тема:** Создание ЭЦП на основе алгоритма DSA с использованием библиотеки OpenSSL.

**Цель работы:** Формирование знаний и навыков по созданию ЭЦП на основе алгоритма DSA с помощью библиотеки OpenSSL.

### Теоретическая часть

DSA (англ. Digital Signature Algorithm — алгоритм цифровой подписи) - криптографический алгоритм с использованием закрытого ключа (из пары ключей: <открытый; закрытый>) для создания электронной подписи, но не для шифрования (в отличие от RSA и схемы Эль-Гамаля). Подпись создается секретно (закрытым ключом), но может быть публично проверена (открытым ключом). Это означает, что только один субъект может создать подпись сообщения, но любой может проверить её корректность. Алгоритм основан на вычислительной сложности взятия логарифмов в конечных полях.

Алгоритм был предложен Национальным институтом стандартов и технологий (США) в августе 1991 и является запатентованным (автор патента - David W. Kravitz), НИСТ сделал этот патент доступным для использования без лицензионных отчислений. DSA является частью DSS (англ. Digital Signature Standard — стандарт цифровой подписи), впервые опубликованного 15 декабря 1998 (документ FIPS-186 (англ. Federal Information Processing Standards — федеральные стандарты обработки информации)). Стандарт несколько раз обновлялся, последняя версия FIPS-186-4.

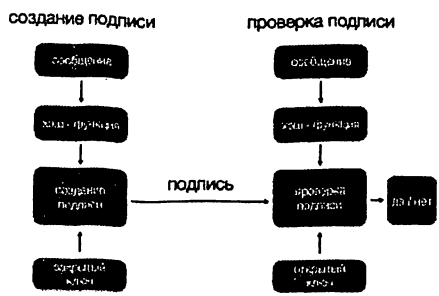



Рис-9.1. Иллюстрация работы DSA

*Параметры схемы цифровой подписи*

Для построения системы цифровой подписи нужно выполнить следующие шаги:

1. Выбор криптографической хеш-функции $H(x)$.
2. Выбор простого числа $q$, размерность которого $N$ в битах совпадает с размерностью в битах значений хеш-функции $H(x)$.
3. Выбор простого числа $p$, такого, что $(p-1)$ делится на $q$. Битовая длина $p$ обозначается $L$ $(2^{\wedge}(L-1)<p<2^{\wedge}L)$.
4. Выбор числа $g$ такого, что его мультипликативный порядок по модулю $p$ равен $q$. Для его вычисления можно воспользоваться формулой $g=h^{\wedge}((p-1)/q) \bmod p$, где $h$ — некоторое произвольное число, $1<h<p-1$ такое, что $g \neq 1$. В большинстве случаев значение $h = 2$ удовлетворяет этому требованию.

Как упомянуто выше, первоочередным параметром схемы цифровой подписи является используемая криптографическая хеш-функция, необходимая для преобразования текста сообщения в число, для которого и вычисляется подпись. Важной характеристикой этой функции является битовая длина выходной последовательности, обозначаемая далее N. В первой версии стандарта DSS рекомендована функция SHA-1 и, соответственно, битовая длина подписываемого числа 160 бит. Сейчас SHA-1 уже не является достаточно безопасной. В стандарте указаны следующие возможные пары значений чисел L и N:

L = 1024, N = 160

L = 2048, N = 224

L = 2048, N = 256

L = 3072, N = 256

В соответствии с этим стандартом рекомендованы хеш-функции семейства SHA-2. Правительственные организации США должны использовать один из первых трех вариантов, центры сертификации должны использовать пару, которая равна или превосходит пару, используемую подписчиками. Проектирующий систему может выбрать любую допустимую хеш-функцию. Поэтому далее не будет заостряться внимание на использовании конкретной хеш-функции.

Стойкость криптосистемы на основе DSA не превосходит стойкость используемой хеш-функции и стойкость пары *(L,N)*, чья стойкость не больше стойкости каждого из чисел по отдельности. Также важно учитывать, как долго система должна оставаться безопасной. В данный момент для систем, которые должны быть стойкими до 2010 (2030) года, рекомендуется длина в 2048 (3072) бита.

*Открытый и секретный ключи*

1. Секретный ключ представляет собой число $0<x<q$
2. Открытый ключ вычисляется по формуле $y=g^{x} \bmod p$

Открытыми параметрами являются числа *(p, q, g, y)*. Закрытый параметр только один — число x. При этом числа (p, q, g) могут быть общими для группы пользователей, а числа x и y являются соответственно закрытым и открытым ключами конкретного пользователя. При подписывании сообщения используются секретные числа x и k, причем число k должно выбираться случайным образом (на практике псевдослучайным) при вычислении подписи каждого следующего сообщения.

Поскольку *(p, q, g)* могут быть использованы для нескольких пользователей, на практике часто делят пользователей по некоторым критериям на группы с одинаковыми *(p, q, g)*. Поэтому эти параметры называют доменными параметрами (Domain Parameters).

## *Практическая часть*

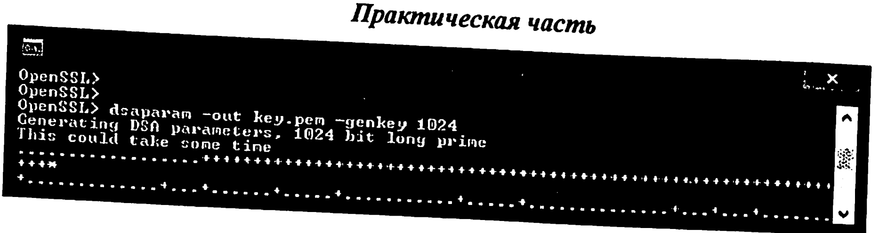



Рис.9.1. Генерация закрытого ключа

```
OpenSSL> dsa -in key.pem -pubout -out public-key.pem
read DSA key
writing DSA key
```

Рис.9.2. Генерация открытого ключа

```
OpenSSL> req -new -key key.pem -out csr.pem
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:UZ
State or Province Name (full name) [Some-State]:Tashkent
Locality Name (eg, city) []:Tashkent
Organization Name (eg, company) [Internet Widgits Pty Ltd]:TUIT
Organizational Unit Name (eg, section) []:Cryptology
Common Name (e.g. server FQDN or YOUR name) []:Otabek
Email Address []:o.o.tursunov@gmail.com

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:123
string is too short, it needs to be at least 4 bytes long
A challenge password []:123
string is too short, it needs to be at least 4 bytes long
A challenge password []:1234
An optional company name []:Turon
```

Рис.9.3. Создание сертификата

```
OpenSSL> dgst -sha256 -sign key.pem -out message.txt.sig message.txt
```

Рис.9.4. Создание электронной цифровой подписи

```
OpenSSL> dgst -sha256 -sign key.pem -out message.txt.sig message.txt
OpenSSL> dsa -in key.pem -pubout -out public-key.pem
read DSA key
writing DSA key
OpenSSL> dgst -sha256 -verify public-key.pem -signature message.txt.sig message.
txt
Verified OK
```

Рис.9.5. Проверка электронной цифровой подписи

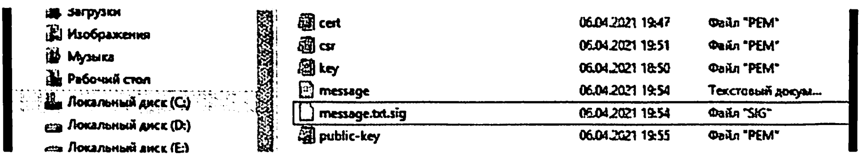



Рис. 9.6. ЭЦП файл

**Задание**

Создать электронную цифровую подпись на основе алгоритма DSA с использованием библиотеки OpenSSL.

**Практическая работа №10**

**Тема:** Создание ЭЦП на основе алгоритма ECDSA с использованием библиотеки OpenSSL.

**Цель работы:** Формирование знаний и навыков по создания ЭЦП на основе алгоритма ECDSA с помощью библиотеки OpenSSL.

**Теоретическая часть**

ECDSA (Elliptic Curve Digital Signature Algorithm) — алгоритм с открытым ключом для создания цифровой подписи, аналогичный по своему строению DSA, но определённый, в отличие от него, не над конечным числовым полем, а в группе точек эллиптической кривой.

Для подписывания сообщений необходима пара ключей — открытый и закрытый. При этом закрытый ключ должен быть известен только тому, кто подписывает сообщения, а открытый — любому желающему проверить подлинность сообщения. Также общедоступными являются параметры самого алгоритма.

*Параметры алгоритма*

1. Выбор хеш-функции $H(x)$. Для использования алгоритма необходимо, чтобы подписываемое сообщение являлось числом. Хеш-функция должна преобразовать любое сообщение в последовательность битов, которые можно потом преобразовать в число.

2. Выбор большого простого числа $q$ — порядок одной из циклических подгрупп группы точек эллиптической кривой.

Замечание: Если размерность этого числа в битах меньше размерности в битах значений хеш-функции $H(x)$ то используются только левые биты значения хеш-функции.

3. Простым числом $p$ обозначается характеристика поля координат $F\{p\}$.

*Генерирование ключей ECDSA*

Для простоты будем рассматривать эллиптические кривые над полем $F\{p\}$, где $F\{p\}$ — конечное простое поле. Причем, если необходимо, конструкцию можно легко адаптировать для эллиптических кривых над другим полем.

Пусть $E$ — эллиптическая кривая, определенная над $F\{p\}$, и $P$ — точка простого порядка $q$ кривой $E(F\{p\})$. Кривая $E$ и точка $P$ являются системными параметрами. Число $p$ — простое. Каждый пользователь — условно назовём его Алиса — конструирует свой ключ посредством следующих действий:

Выбирает случайное или псевдослучайное целое число $x$ из интервала $[1, q\text{-}1]$.

Вычисляет произведение (кратное) $Q = x \cdot P$.

Открытым ключом пользователя Алисы $A$ является точка $Q$, а закрытым -$x$.

Вместо использования $E$ и $P$ в качестве глобальных системных параметров, можно фиксировать только поле $F\ \{p\}$ для всех пользователей и позволить каждому пользователю выбирать свою собственную эллиптическую кривую $E$ и точку $P$ in $E(F\{p\})$. В этом случае определенное уравнение кривой $E$, координаты точки $P$, а также порядок $q$ этой точки $P$ должны быть включены в открытый ключ пользователя. Если поле $F\{p\}$ фиксировано, то аппаратная и программная составляющие могут быть построены так, чтобы оптимизировать вычисления в том поле. В то же время имеется огромное количество вариантов выбора эллиптической кривой над полем $F\{p\}$.

*Вычисление цифровой подписи*

Для того, чтобы подписать какое-либо сообщение, для которого подсчитано значение $h$ хеш-функции $H$, пользователь $A$ должен сделать следующее:

*1*. Выбрать случайное целое число $k \in [1, q\text{-}1]$.

2. Вычислить $k * P = (x_1, y_1)$ и положить в $r = x_1 \ mod\ q$, где $r$ получается из целого числа $x_1$ между 0 и $(p - 1)$ приведением по модулю $q$.

Замечание: $r$=0, то уравнение подписи s = $k^{-1}\,(h + x * r) \ mod\ q$ не зависит от секретного ключа $x$, и следовательно, $(r, s)$ не подходит в качестве цифровой подписи. Значит, в случае $r = 0$ необходимо вернуться к шагу 1. Если над кривой нет точки с абсциссой 0, можно пропустить эту проверку.

3. Вычислить $k^{-1}$ $mod\, q$ и положить $s = k^{-1}\,(h + x * r)\, mod\, q$, где h — значение хеш-функции подписываемого сообщения.

Замечание: если $s$=0, то значение $s^{-1}\, mod\, q$, нужное для проверки, не существует. Значит, в случае $s = 0$ необходимо вернуться к шагу 1. Однако этот случай очень редкий.

Подписью для сообщения является пара целых чисел ($r$, $s$).

*Проверка цифровой подписи*

Для того, чтобы проверить подпись пользователя Алисы ($r$, $s$) на сообщение, пользователь Боб $B$ должен сделать следующее:

1. Получить подтвержденную копию открытого ключа $Q$ пользователя $A$;
2. Проверить, что числа $r$ и $s$ являются целыми числами из интервала [1, q-1], и вычислить значение хеш-функции $h$ от сообщения;
3. Вычислить $u_1 = s^{-1} * h\, mod\, q$ и $u_2 = s^{-1} * r\, mod\, q$; это возможно так как $s \neq 0$.
4. Вычислить $u_1 * P + u_2 * Q = (x_0, y_0)$, и относительно $x_0$, как целого числа между 0 и ( $p - 1$ ), положить $v = x_0\, mod\, q$;
5. Принять подпись, тогда и только тогда, когда $v = r$.

Заметим, что, если пользователь Алиса вычислила свою подпись правильно, то $u_1 * P + u_2 * Q = (u_1 + x * u_2) * P = (s^{-1} * h\, mod\, q\ + x * s^{-1} * r\, mod\, q)$ так как $k = s^{-1}(h + x * r) mod\, q$ и, и поэтому $v = r$.

Для подтверждения публичного ключа Q нужно проделать следующее ($O$ здесь обозначает бесконечно удалённую точку):

Проверить, что $Q$ не равно $O$ и координаты верны;

Проверить, что $Q$ лежит на кривой;

Проверить, что $q$*$Q = O$;

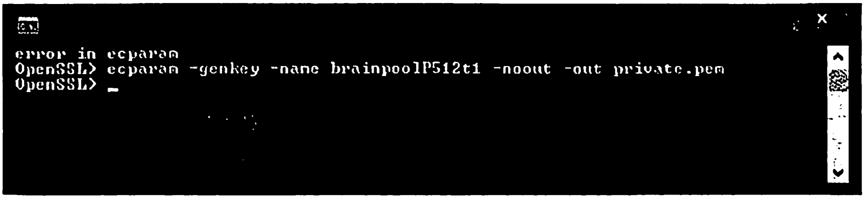



Рис.10.1. Генерация закрытого ключа

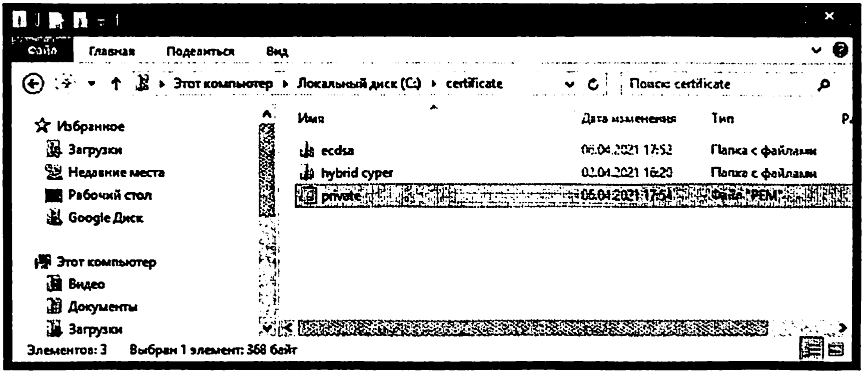



Рис.10.2. Файл закрытого ключа

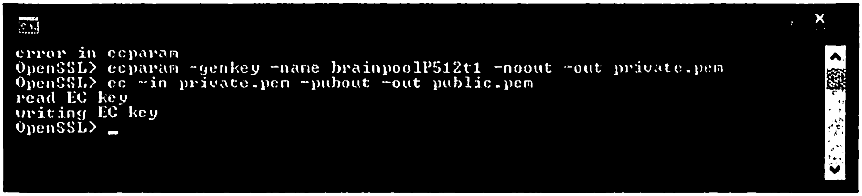



Рис.10.3. Генерация открытого ключа

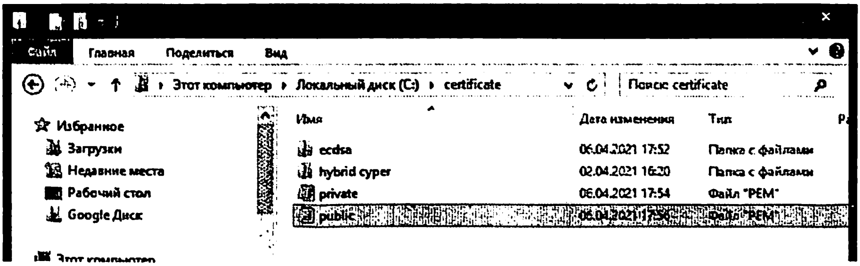



Рис.10.4. Файл закрытого ключа

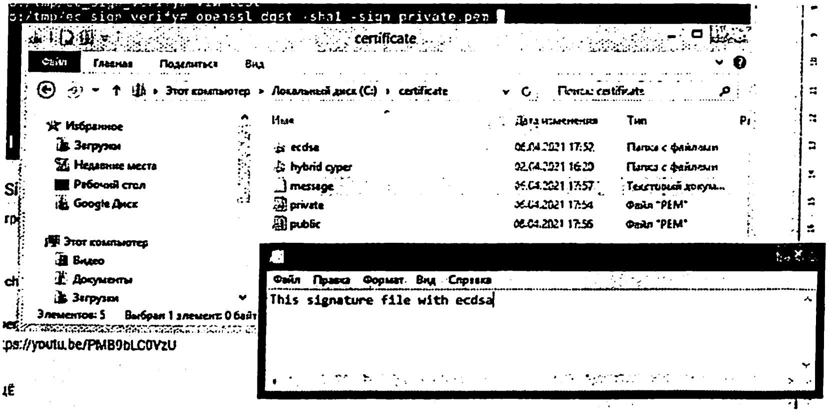



Рис.10.5. Создание открытых данных

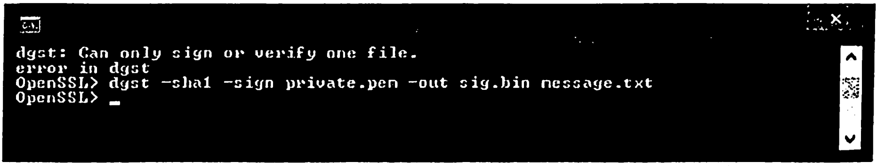



Рис.10.6. Создание электронной цифровой подписи

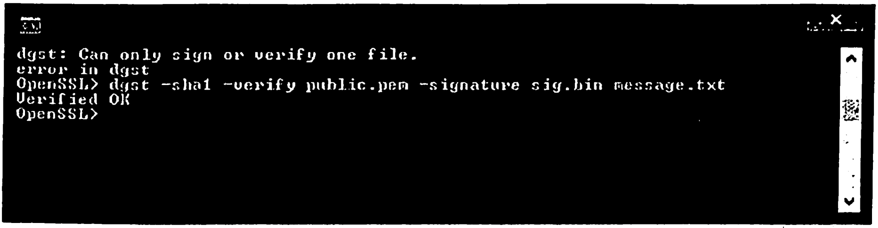



Рис.10.7. Проверка электронной цифровой подписи

## Задание

Создать электронную цифровую подпись на основе алгоритма ECDSA с использованием библиотеки OpenSSL.

## Практическая работа № 11

**Тема:** Создание сертификата X.509 с использованием библиотеки OpenSSL.

**Цель работы:** Формирование знаний и навыков по созданию сертификата X.509 на основе алгоритма DSA с помощью библиотеки OpenSSL.

### Теоретическая часть

X.509 — стандарт ITU-T (англ. International Telecommunication Union — Telecommunication sector) для инфраструктуры открытого ключа (англ. Public key infrastructure, PKI) и инфраструктуры управления привилегиями (англ. Privilege Management Infrastructure).

X.509 определяет стандартные форматы данных и процедуры распределения открытых ключей с помощью соответствующих сертификатов с цифровыми подписями. Эти сертификаты предоставляются удостоверяющими центрами (англ. Certificate Authority). Кроме того, X.509 определяет формат списка аннулированных сертификатов, формат сертификатов атрибутов и алгоритм проверки подписи путём построения пути сертификации. X.509 предполагает наличие иерархической системы удостоверяющих центров для выдачи сертификатов.

*Структура сертификата X.509*

- Сертификат
- Версия
- Серийный номер
- Идентификатор алгоритма подписи
- Имя издателя
- Период действия:
    - Не ранее
    - Не позднее
- Имя субъекта
- Информация об открытом ключе субъекта:

- Алгоритм открытого ключа
- Открытый ключ субъекта
- Уникальный идентификатор издателя (обязательно только для v2 и v3)
- Уникальный идентификатор субъекта (обязательно только для v2 и v3)
- Дополнения (для v2 и v3)
  - Возможные дополнительные детали
- Алгоритм подписи сертификата (обязательно только для v3)
- Подпись сертификата (обязательно для всех версий)


Рис.11.1 Общий стандарт для интернета, использующий формат X.509

В 4 разделе RFC 2459 описана структура сертификата X.509 v3. Обновленная структура сертификата X.509 v3 описана в документе RFC 5280 в 4 разделе. Она специализирована под интернет-приложения.

Для описания внутренней структуры сертификатов X.509 используется ASN.1. Хранятся, как правило, в виде DER или PEM-файлов. Общепринятое расширение .cer или .crt. Может быть и другое расширение.

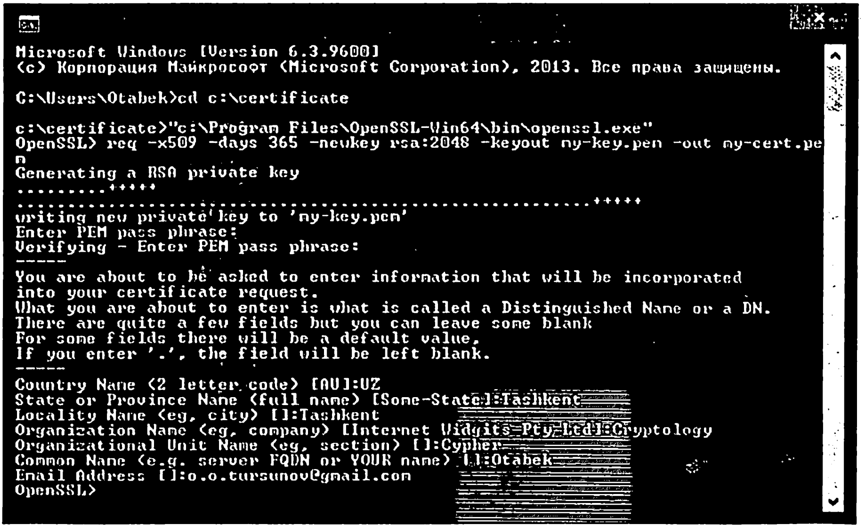



Рис 11.1. Создание закрытого ключа и подписание сертификата X.509

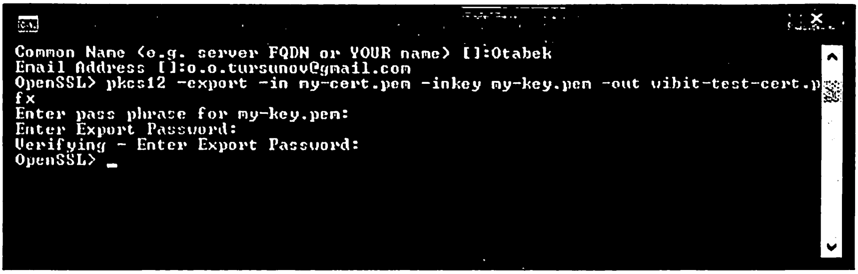
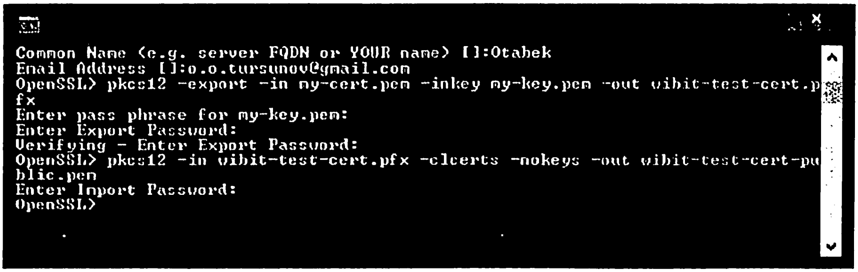



Рис 11.2. Преобразование сертификата в файл pfx.




(2)

Рис 11.3. Генерация открытого ключа из файла pfx

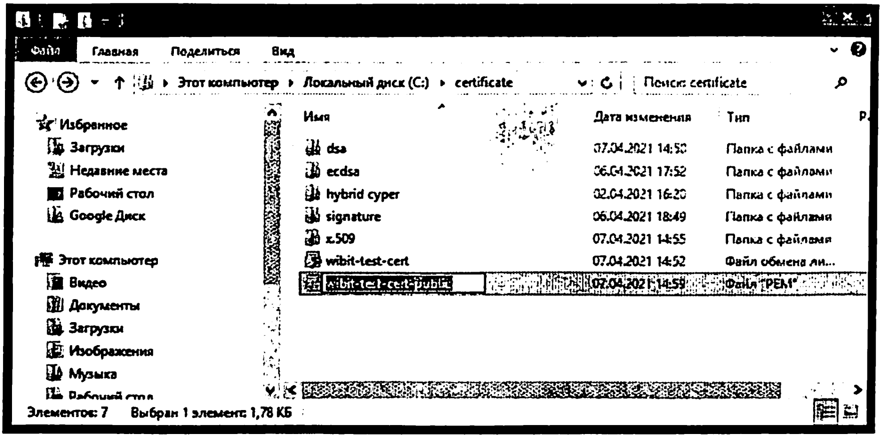



Рис.11.4. Файл открытого ключа

## Задание

Создать сертификат X.509, сделать снимки экрана процесса создания сертификата, подготовить отчёт с этими снимками.

## Практическая работа № 12

**Тема:** Сканирование и анализ потока SSL / TLS

**Цель работы:** Формирование знаний и навыков работы с протоколом SSL / TLS, расшифровка пакетов протокола SSL / TSL.

### Теоретическая часть

Wireshark – это широко известный и бесплатный инструмент для анализа сети.

Первый шаг в его использовании – загрузить его отсюда и установить.

Еще одна вещь, которую вам нужно сделать перед расшифровкой TLS-зашифрованного трафика, – это настроить ваш веб-браузер для экспорта клиентских TLS-ключей.

Поскольку TLS предназначен для защиты конфиденциальности клиента и сервера во время передачи, логично, что он разработан таким образом, что любой из них может расшифровать трафик, а никто другой не может.

Поскольку мы действуем в качестве подслушивающего устройства в сети (именно это и предотвращает TLS), нам необходимо, чтобы одна из доверенных сторон поделилась с нами своими секретами.

В Firefox и Chrome это можно сделать, установив переменную среды **SSLKEYLOGFILE**.

Если эта переменная установлена, оба браузера настроены на сохранение копии секретов клиента в указанном месте.

В Linux эту переменную можно установить с помощью команды Export.

В Windows его можно установить, открыв «Дополнительные параметры системы», выбрав «Переменные среды» и добавив новую системную переменную.

Пример этой переменной в Windows показан ниже:

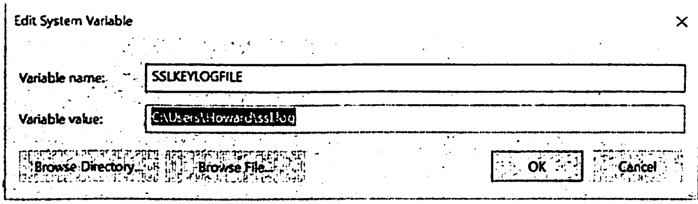



Рис. 12.1. Переменная в Windows

После установки переменной среды рекомендуется перезагрузить систему, чтобы убедиться, что новые параметры активны.

После этого у нас есть все, что нужно для расшифровки трафика TLS.

*Выполнение расшифровки трафика*

Если вы хотите расшифровать трафик TLS, вам сначала нужно его перехватить.

По этой причине важно, чтобы Wireshark был запущен до начала сеанса просмотра веб-страниц.

Прежде чем мы начнем захват, мы должны подготовить его для расшифровки трафика TLS.

Для этого нажмите **Edit → Preferences**. Выберите **Protocols** в левой панели и прокрутите вниз до TLS.

В этот момент вы должны увидеть что-то похожее как на экране ниже.

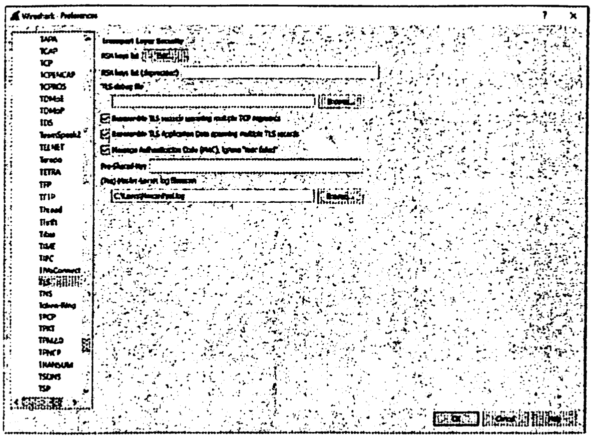


Рис. 12.2. Протоколы в wireshark

Внизу этого экрана есть поле для (Pre) -Master-Secret имени файла журнала.

Как показано выше, вам нужно установить это значение в том же месте, что и SSLKEYLOGFILE для вашего браузера. Когда закончите, нажмите ОК.

Теперь на главном экране Wireshark будет показан список возможных адаптеров для захвата.

В этом примере я буду использовать WiFi 2, так как по нему проходит трафик (показан черной линией)

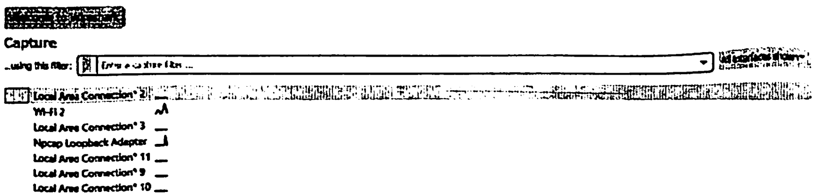



Рис. 12.3. Интерфейсы, к которым подключено устройство

При нажатии на адаптер начнется захват трафика на этом устройстве.

Теперь вы готовы создать некоторый трафик с шифрованием TLS.

Перейдите в Chrome или Firefox и перейдите на сайт, который использует HTTPS (мы использовали Facebook для этого примера).

После загрузки вернитесь в Wireshark и остановите захват (красный квадрат).

Просматривая снимок, вы, вероятно, увидите много трафика.

Ищем пакеты, связанные с вашим сеансом просмотра в TLS-шифровании.

Одним из способов является поиск DNS и фильтрация по предоставленному IP-адресу (показано ниже).

На изображении ниже показан пакет из нашего сеанса просмотра в Facebook.

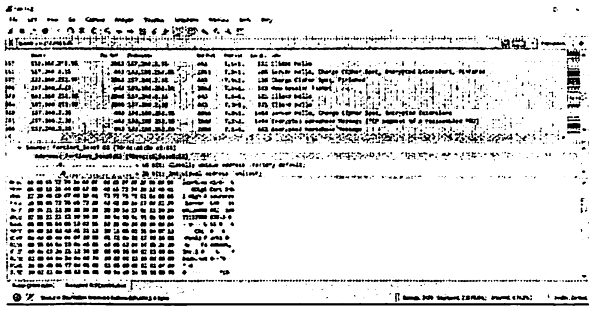


Рис. 12.4. Пакет из сеанса в Facebook

Как видно, Wireshark показывает несколько разных вкладок в нижней части окна.

В дополнение к вкладке Frame, один помечен как расшифрованный TLS.

Глядя на ASCII-представление пакета, мы видим сертификат веб-сайта (включая слово Facebook).

На данный момент мы успешно расшифровали трафик TLS в Wireshark.

*Дешифровка TLS с помощью wireshark.*

Для расшифровки TLS трафика в современных браузерах появилась возможность сохранять сеансовые ключи. Для этого используется переменная окружения SSLKEYLOGFILE. Проще всего в настройку окружений попасть выполнив команду systempropertiesadvanced и нажав Environment Variables. Назначить в ней путь до файла, браузер будет сохранять в него сеансовые ключи.

SSLKEYLOGFILE = C:\Users\username\sslkeylog.log

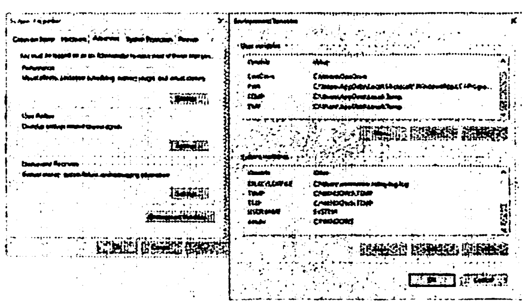


Рис. 12.5. Файл SSLKEYLOGFILE

После этого настраиваем Wireshark на чтение этого файла. Заходим в Edit -> Preferences -> Protocols -> SSL -> Pre-Master-Secret log filename

И указываем путь до этого файла.

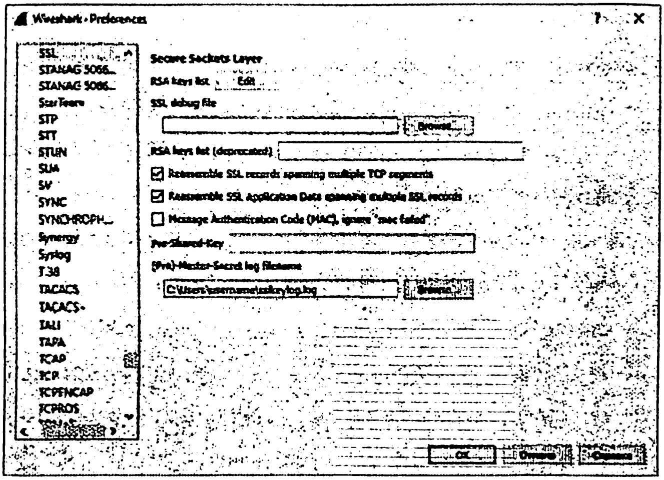



Рис. 12.6. Указание пути к файлу

После этого зашифрованный трафик будет виден.

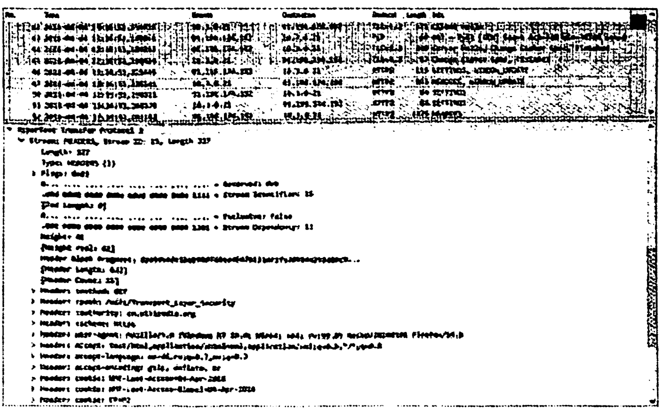


Рис. 12.7. Вид зашифрованного трафика

## Задание

Установить wireshark, отследить и анализировать протокол SSL / TLS.

## Практическая работа № 13

**Тема:** Программная проверка гомоморфных свойств криптосистем с открытым ключом

**Цель работы:** Формирование знаний и навыков по гомоморфного шифрования, использование гомоморфного шифрования.

### Теоретическая часть

Гомоморфное шифрование — форма шифрования, позволяющая производить определённые математические действия с зашифрованным текстом и получать зашифрованный результат, который соответствует результату операций, выполненных с открытым текстом. Например, один человек мог бы сложить два зашифрованных числа, не зная расшифрованных чисел, а затем другой человек мог бы расшифровать зашифрованную сумму — получить расшифрованную сумму, не имея расшифрованных чисел.

Различают криптосистемы частично гомоморфные и полностью гомоморфные. Частично гомоморфная криптосистема позволяет производить только одну из операций — либо сложение, либо умножение. Полностью гомоморфная криптосистема поддерживает выполнение обеих операций, то есть, в ней выполняются свойства гомоморфизма как относительно умножения, так и относительно сложения.

*Общий вид гомоморфного шифрования*

Гомоморфное шифрование является формой шифрования, позволяющей осуществить определённую алгебраическую операцию над открытым текстом посредством выполнения алгебраической операции над зашифрованным текстом.

Пусть $k$ — ключ для шифрования, $t$ — подлежащий шифрованию открытый текст (сообщение), $E(k,t)$ — выполняющая шифрование функция.

Функция $E$ называется гомоморфной относительно операции «*» (сложения или умножения) над открытыми текстами (сообщениями) $t_1$ и $t_2$, если существует эффективный алгоритм $M$ (требующий полиномиального числа ресурсов и работающий за полиномиальное время), который, получив на вход любую пару шифрованных текстов вида $E(k,t_1)$ и $E(k,t_2)$, выдаёт шифрованный текст (шифротекст, криптограмму) $c = M(E(k,t_1))$такой, что при расшифровании $c$ будет получен открытый текст $t_1 * t_2$.

На практике чаще рассматривается следующий частный случай гомоморфного шифрования.

Пусть для данной функции шифрования $E$ и операции «$*_1$» над открытыми текстами $t_1$ и $t_2$ существует операция «$*_2$» над шифрованными текстами, такая, что из шифрованного текста $c = M(E(k,t_1))$ при его расшифровании извлекается открытый текст $t_1 * t_2$. При этом требуется, чтобы по заданным $c$, $c = E(k,t_1) *_2 E(k,t_2)$, но при неизвестном ключе $k$, было бы невозможно эффективно проверить, что шифрованный текст $c$ получен из $E(k,t_1)$ и $E(k,t_2)$

Любую стандартную систему шифрования можно описать, описав три операции: операцию генерация ключей (*keyGen*), операцию шифрования (*encypt*) и операцию расшифрования (*decrypt*).

Для описания гомоморфной системы шифрования кроме трёх перечисленных выше операций нужно описать операцию вычислений (*eval*). Использование гомоморфного шифрования подразумевает использование последовательности из четырёх операций: генерации ключей, шифрования, вычисления, расшифрования:

1. генерация ключей — генерирование клиентом открытого ключа (*public key*) ***pk*** (для расшифрования зашифрованного открытого текста) и секретного ключа (*secret key*) ***sk*** (для шифрования открытого текста);

2. шифрование — шифрование клиентом открытого текста (*plain text*) ***PT*** с использованием секретного ключа ***sk*** — вычисление шифрованного текста (*cipher text*) $CT{=}E_{sk}(PT)$; отправка клиентом шифрованного текста ***CT*** и открытого ключа ***pk*** на сервер;

3. вычисление — получение сервером функции ***F***, использование ***F*** и ***pk*** для выполнения вычислений над шифрованным текстом ***CT***; отправка сервером результата клиенту;

4. расшифрование — расшифрование клиентом полученного от сервера значения с использованием $sk$.

Пусть $\boldsymbol{E}$ — функция шифрования; $\boldsymbol{D}$ — функция расширования; $t_1$ и $t_2$ — открытые тексты; символы «$\otimes$» и «$\otimes$» обозначают операции умножения и сложения над шифрованными текстами, соответствующие операциям умножения и сложения над открытыми текстами.

Система шифрования является гомоморфной относительно операции умножения (обладает мультипликативными гомоморфными свойствами), если $D(E(t_1) \otimes E(t_2)) = t_1 + t_2$.

Система шифрования является гомоморфной относительно операции сложения (обладает аддитивными гомоморфными свойствами), если $D(E\ (t_1) \otimes E(t_2)) = t_1 + t_2$.

Система шифрования является гомоморфной относительно операций умножения и сложения, то есть, полностью гомоморфной (обладает и мультипликативными, и аддитивными гомоморфными свойствами), если $D(E\ (t_1) \otimes E(t_2)) = m_1 + m_2, D(E(m_1) \otimes E(m_2)) = m_1 + m_2$

Если криптосистема с такими свойствами сможет зашифровать два бита, то, поскольку операции сложения и умножения формируют над битами полный по Тьюрингу базис, становится возможным вычислить любую булеву функцию, а следовательно, и любую другую вычислимую функцию.

*Частично гомоморфные системы*

Частично гомоморфные криптосистемы — это такие криптосистемы, которые гомоморфны относительно только одной операции — либо операции сложения, либо операции умножения. В приведённых ниже примерах выражение $E(t)$ обозначает использование функции шифрования $E$ для шифрования открытого текста (сообщения) $t$.

*Криптосистема RSA*

Криптосистема RSA является криптографической схемой с открытым ключом, гомоморфной по умножению. Пусть $n$ — модуль RSA, $t$ — открытый текст, $k$ — открытый ключ (для шифрования открытого текста). Функция шифрования имеет $E(t) = t^k \bmod n$. Покажем гомоморфизм по умножению: $E(t_1) * E(t_2) = {t_1}^k * {t_2}^k \bmod n = t_1 {t_2}^k \bmod n = E(t_1 t_2)$.

*Криптосистема Эль-Гамаля*

Криптосистема Эль-Гамаля является альтернативой криптосистемы RSA и при равном значении ключа обеспечивает ту же криптостойкость. Эль-Гамаль усовершенствовал алгоритм Диффи — Хеллмана и получил алгоритмы для шифрования и для обеспечения аутентификации. Криптосистема является криптосистемой вероятностного шифрования. Её функция шифрования гомоморфна относительно операции умножения открытых текстов: криптограмма произведения может быть вычислена как произведение (попарное) криптограмм сомножителей. Пусть $E$ — функция шифрования; $t_1$ и $t_2$ — открытые тексты. Если $E(y, g, \{r_1\}, t_1) = (y^{r_1} t_1, g^{r_1})$ и $E(y, g, \{r_2\}, t_2) = (y^{r_2} t_2, g^{r_2})$ можно получить в виде $(y^{r_1} y^{r_2} t_1 t_2, g^{r_1} g^{r_2})$.

*Вполне гомоморфное шифрование*

Генерация ключа: выбираем $p$=99

Шифрование двух битов 1:

выбираем $q_1 = 37$ и $q_2 = 82$, а также $r_1 = 3$, и $r_2 = 2$.

вычисляем $c_1 = q_1 * p + 2 * r_1 + 1 = 3670, c_2 = q_2 * p + 2 * r_2 + 1 = 8123$.

Вычисление суммы 1⊕1 и произведения 1×1: $c_1 + c_2 = 11793$, $c_1 \times c_2 =$ 29811410.

Расшифрование результата:

(11793 mod 99) mod 2 =4 mod 2 = 0 = 1⊕1

(9032270 mod 99) mod 2 =35 mod 2 = 1=1×1.

**Задание**

Зашифровать и расшифровать свою фамилию и имя с помощью гомоморфного шифрования на основе алгоритма RSA.

## Список литературы


1. Katz J., Lindell Y. Introduction to modern cryptography. – CRC press, 2014.

2. Stamp M. Information security: principles and practice. – New York : Wiley, 2011. – Т. 2.

3. Akbarov D.E. Axborot xavfsizligini ta'minlashning kriptografik usullari va ularning qo'llanilishi. Toshkent, 2008 – 394 bet.

4. G'aniev S. K., Karimov M. M., Tashev K. A. Axborot xavfsizligi. Axborot-kommunikatsiya tizimlar xavfsizligi. Oliy o'quv yurt talabalari uchun mo'ljallangan. "Aloqachi", 2008.

5. Шнайер Б. Прикладная криптография. Протоколы, алгоритмы, исходные тексты на языке Си //М.: Триумф. – 2002. – Т. 816. – С. 3.

# СОДЕРЖАНИЕ

Рассмотрено и рекомендовано к печати
на заседании учебно-методического совета
ТУИТ им.Мухаммада ал-Хоразмий
2021 год 25 Мая
протокол №11(135)

Составители: О.О. Турсунов
Н.Ф. Ахмедова
Ш.З. Исломов

Рецензенты: к.т.н. Ж.Д. Мукимов
PhD. Ф.Б. Ботиров

Главный редактор: ____________

Корректировщики: ____________

**Формат 60x84 1/16. Печ.лист 4,25.**
**Заказ №75. Тираж 10.**
**Отпечатано в «Редакционно издательском» отделе при ТУИТ.**
**Ташкент ул. Амир Темур, 108.**