Министерство образования Республики Беларусь

Учреждение образования
«Полоцкий государственный университет»

Д.Ф. Пастухов
Ю.Ф. Пастухов
П.Р. Синица

Шифрование данных на базе эллиптических кривых

Учебно-методическое пособие к лекционным и практическим занятиям
для студентов специальности
1-98 01 01 Компьютерная безопасность

Новополоцк
ПГУ
2016

УДК 004.94

Одобрено и рекомендовано к изданию
методической комиссией факультета информационных технологий
В качестве учебно-методического пособия

Кафедра технологий программирования

Рецензенты:
А.Ф. Оськин, кандидат технических наук, доцент, доцент кафедры технологий программирования;
Р.П. Богуш, кандидат технических наук, доцент, заведующий кафедрой вычислительных систем и сетей

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Шифрование данных методом эллиптических кривых преследует цели выработать метод быстрого и эффективного шифрования на базе эллиптической криптографии и в то же время повысить устойчивость шифрования (стойкость шифра) и целостность передаваемой информации в процессе протоколе обмена данными.

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 неизвестно существование субэкспоненциальных алгоритмов решения задачи дискретного логарифмирования. Использование эллиптических кривых для создания криптосистем было независимо предложено Нилом Коблицем и Виктором Миллером в 1985 году.

Нилом Коблицем и Виктором Миллером было предложено использовать в криптографии алгебраические свойства эллиптических кривых. Роль основной криптографической операции выполняет операция скалярного умножения точки на эллиптической кривой на данное целое число, определяемое через операции сложения и удвоения точек эллиптической кривой. Последние, в свою очередь, выполняются на основе операции сложения, умножения и инвертирования в конечном поле, над которыми рассматривается кривая.

Особый интерес к криптографии эллиптических кривых обусловлен теми преимуществами, которые дают её применение в беспроводных коммуникациях - высокое быстродействие и небольшая длина ключа. Асимметрическая криптография основана на сложности решения некоторых математических задач. При использовании алгоритмов на эллиптических кривых предполагается, что не существует субэкспоненциальных алгоритмов для решения задач дискретного логарифмирования в группах их точек. При этом порядок группы точек э.к. определяет сложность задачи.

Например, на конференции RSA 2005, Агенство национальной безопасности объявило о создании "Suite B", в котором используется исключительно алгоритмы эллиптической криптографии, причём для защиты информации классифицируемой "Top Secret" используются всего лишь 384 – битные ключи.

Эллиптической кривой называется множество точек (x, y), удовлетворяющих уравнению:

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$$

Это уравнение может рассматриваться над произвольными полями и, в частности над конечными полями, представляющими для криптографии особый интерес.

В криптографии эллиптические кривые рассматриваются над двумя типами конечных полей: простыми полями нечётной характеристики $(Z_p, p > 3 - простое число)$ и полями характеристики 2 $(GF(2^m))$.

Для использования эллиптической криптографии все участники должны согласовать все параметры, определяющие эллиптическую кривую , т.е. набор параметров криптографического протокола. Эллиптическая кривая определяется константами a, b, p. Абелева подгруппа точек является циклической и задаётся одной порождающей точкой G. Итак, для конечного поля $Z_p, p > 3$ необходимо задать набор параметров (p, a, b, G, n). Где n - порядок точки образующего элемента G. Существует несколько рекомендованных наборов параметров:

- NIST
- SECG

Для создания собственного набора необходимо:

1) Выбрать набор параметров
2) Найти эллиптическую кривую, удовлетворяющих набору параметров.

Для нахождения кривой для заданного набора параметров используют два метода:

- Выбрать случайную кривую, затем использовать алгоритм подсчёта точек.
- Выбрать точки, после чего построить кривую по этим точкам, используя технику умножения.

NIST рекомендует 15 эллиптических кривых, многие из которых были получены Jerry Solinas , некоторые из них:

- поля F_p ,где простое число p имеет длину 192,224,256,384 или 521 бит.
- поля F_{2^m} , где $m = 163,233,283,409,571$.

Размер ключа

Размер поля должен как минимум в 2 раза превосходить размер ключа. Например, для 128 битного ключа рекомендуется использовать эллиптическую кривую над полем F_p, где p имеет длину 256 бит.

Область применения эллиптической криптографии.

В современных отечественных стандартах формирования и проверки ЭЦП (электронной цифровой подписи) ГОСТ Р 34.10-2001 и ГОСТ Р 34.10.2012

также применяются алгоритмы на эллиптических кривых ,стойкость которых основывается на сложности вычисления дискретного логарифма в группе точек эллиптической кривой, а также на стойкости хеш – функции . Эллиптические кривые применяются в современных системах:

1) Информационные системы организаций крупного бизнеса. Предприятия крупного бизнеса заинтересованы в большей степени в защите своей коммерческой тайны. В связи с этим вопросы цены в таком случае уходят на второй план. Здесь целесообразно применение сертифицированных средств защиты информации, таких как программный комплекс CSP VPN.
2) Информационные системы организации среднего и малого бизнеса, например, идентификаторы RuToken ЭЦП, eToken ГОСТ.
3) Мобильная торговля. В данной сфере распространено применение различных протоколов передачи данных, например, протокол беспроводной передачи данных WAP в сотовых телефонах, карманных компьютерах и т.д.
4) Информационные системы государственных учреждений. Применяют различные сертификационные комплексы ЗАСТАВА, CPN VPN Server.
5) Операции в банковских учреждениях.
6) Интернет – приложения. В данном случае распространено применение криптографических протоколов с алгоритмами на эллиптических кривых, например, Secure Sockets Layer (SSL) – криптографический протокол защищённости сокетов.

Безопасность эллиптической криптографии

Безопасность, обеспечиваемая подходом на основе эллиптических кривых, зависит от того насколько трудной является задача определения k по данным kP и P. Эту задачу называют проблемой логарифмирования на эллиптической кривой. Логарифмирование на эллиптической кривой с помощью метода Полларда составляет

Размер ключа	MIPS - годы
150	$3.8*10^{10}$
205	$7.1*10^{18}$
234	$1.6*10^{28}$

Эллиптические кривые используют для построения цифровой электронной подписи. Алгоритм ECDSA(Elliptic Curve Digital Signature Algorithm) принят в качестве стандартов ANSI X9F1 и IEEE P1363.Перечислим преимущества эллиптической криптографии:

1) Сравнительно меньшая длина ключа
2) Скорость работы эллиптических алгоритмов гораздо выше, чем у классических. Это объясняется как размерами поля, так и применением более близкой для компьютеров структуры бинарного конечного поля.
3) Из-за маленькой длины ключа и высокой скорости работы алгоритмы ассиметричной криптографии на эллиптических кривых могут использоваться в смарт – картах и в других устройствах с ограниченными вычислительными возможностями.

Секретность и стойкость (устойчивость по отношению к атакам злоумышленников) шифрования приведенного алгоритма согласно структурной схеме заключается несколькими важными характеристиками, введёнными в алгоритм:

1) Использование группы точек эллиптической кривой на базе изученных кривых алгебраической геометрией увеличивают скорость шифрования, дешифрования а, главное, стойкости шифра. Данные преимущества обнаруживаются при сравнении шифра эллиптической кривой с ключом одинаковой длины по сравнению с шифрами только на базе операций в конечных полях. Таких как, в криптосистемах с открытым ключом, криптосистеме без передачи ключей, криптосистеме с электронной подписью. Эти системы используют конечные поля с характеристикой p (с операциями над целыми числами, возведение в целую степень по mod p), секретность которых основана на использовании функции Эйлера и теоремы Эйлера – Ферма. Групповая операция произведения двух чисел заменяется на групповую операцию сложения чисел, что значительно уменьшает вычислительную сложность алгоритмов и вероятность ошибки вычислений при работе с большими целыми числами, так как величины всех чисел не превосходят характеристики поля p.
2) Секретность шифра заключается в использовании хеш – функции случайного числа, которое по каналу связи может не передаваться. Несимметричность канала связи и протокола обмена данными удовлетворяет всем современным требованиям стойкости шифрования в эллиптической криптографии.
3) Секретность шифра заключается в уникальности алфавитной строки, перебор которых потребует порядка 10^{30} лет из расчёта перебора 10^{9} строк алфавита в 1 с. Это при условии, что злоумышленник знает все параметры эллиптической кривой и даже образующий элемент группы G.
4) Простота выполнения сложения двух точек эллиптической кривой, работа со сравнительно небольшими числами (по сравнению

с их умножением двух чисел) значительно увеличивает скорость шифрования и дешифрования. В то время как стойкость шифра на основе эллиптической криптографии значительно выше, чем для методов с применением теоремы Эйлера – Ферма, так как решение обратной задачи в случае эллиптического шифрования значительно сложнее при одинаковом размере ключа, что связано с групповыми структурами в алгебраической геометрии.

Постановка задачи

Эллиптические кривые

В криптографических методах используют эллиптические кривые над полем целых чисел с характеристикой поля $r = 2$ либо более $r > 3$. В дальнейшем мы будем рассматривать поле целых чисел с характеристикой $r > 3$.

Криптографические кривые с характеристикой поля $r > 3$ имеют канонический вид:

$$y^2 = x^3 + ax + b \qquad (1)$$

Где a, b – целочисленные коэффициенты кривой, p – простое достаточно большое число.

Как видно из формулы (1), если точка с координатами (x, y) удовлетворяет уравнению (1), то уравнению (1) удовлетворяет также и точка с $(x, -y)$. Под э.к. понимают геометрическое множество точек (1) дополненное бесконечно – удалённой точкой.

Следующее число, называемое дискриминантом кривой $\Delta = -16\left(4a^3 + 27b^2\right)$, не должно быть равным нулю (в этом случае отсутствуют точки самопересечения и точки возврата). Если дискриминант положителен $\Delta > 0$, график кривой имеет 2 части, если $\Delta < 0$, то одну часть.

На множестве точек эллиптической кривой определяют группу по сложению точек э. к. (раздел математики называется алгебраической геометрией). Суммой двух точек э. к. P, Q называется третья точка R, лежащая на прямой PQ и эллиптической кривой одновременно, и обозначается как $R = P + Q$, т.е. $-R + P + Q = 0$.

Операцией группового сложения называют 3 точки э. к., удовлетворяющих уравнению:

$$R' + P + Q = 0 \qquad (2)$$

Откуда видно, что $R' = -R$ (R', R – элементы взаимно обратные по групповой операции). С другой стороны, прямая параллельная координатной оси у, пересекает ровно 2 точки э.к. (зеркально симметричные относительно оси х) и бесконечно удалённую точку, следовательно, взаимно обратные точки э.к R', R – имеют координаты (x, y) и $(x, -y)$ соответственно . Единицей по групповому сложению определяют геометрически бесконечно удалённую

точку и обозначают 0. Итак, для групповой операции по сложению необходимо провести секущую через точки P,Q и зеркально отобразить точку R, $R^{'} = -R$.

Возможны частные случаи:

1) $P = Q$ - секущая прямая вырождается в касательную $R^{'} + 2P = 0$

2) $P + Q + 0 = 0 \Leftrightarrow P = -Q$ точки P,Q имеют одинаковые абсциссы. Следующей точкой по сложению выбирают $Q + 0 = Q$.

3) $P + P + 0 = 0 \Leftrightarrow P = 0$ - секущая прямая одновременно является вертикальной прямой и касательной.

Криптография использует конечные циклические абелевые группы с порождающим элементом G. При этом любую точку э.к. циклической группы $1 \le k \le n0$ получают по формуле $P_k = (GG\cdots G)$. Порядком группы точек э.к. называется число $n0$, такое что $P_{n0} = O$ - нулевой элемент группы. Зная порождающий элемент группы G, можно составить таблицу всех точек эллиптической кривой, при сложении точек с порядком $k > n0$ все точки периодически повторяются $P_k = P_{k-n0*s}$, где $1 \le k - n0 * s \le n0 - 1, s \in N$.

В зависимости от общей ситуации и частных случаев 1),2),3) координаты точек э.к. вычисляют по формулам (индексы 1 и 2 соответствуют точкам P,Q соответственно):

$$\begin{cases} x = \left(\dfrac{y_2 - y_1}{x_2 - x_1}\right)^2 - x_1 - x_2 = k^2 - x_1 - x_2 \\ y = -y_1 + \left(\dfrac{y_2 - y_1}{x_2 - x_1}\right)(x_1 - x) = -y_1 + k\left(2x_1 + x_2 - k^2\right) \end{cases} \quad (3)$$

$$\begin{cases} x = \left(\dfrac{3x_1^2 + a}{2y_1}\right)^2 - 2x_1 = k^2 - 2x_1 \\ y = -y_1 + \left(\dfrac{3x_1^2 + a}{2y_1}\right)(x_1 - x) = -y_1 + k\left(3x_1 - k^2\right) \end{cases} \quad (4)$$

Вывод (формул (3),(4)):

Угловой коэффициент прямой проходящий через 2 точки равен $k = \dfrac{y_2 - y_1}{x_2 - x_1} = \dfrac{y - y_1}{x - x_1}$, где точка прямой (x, y) является скользящей по прямой (переменной). Для точек 1 (x_1, y_1) , 2 (x_2, y_2), (x, y) получим

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

$y_1^{\,2} = x_1^{\,3} + ax_1 + b$,

$y_2^{\,2} = x_2^{\,3} + ax_2 + b$

Вычтем $(y_2 - y_1)(y_2 + y_1) = (x_2 - x_1)\left(x_2^2 + x_1x_2 + x_1^2\right) + a(x_2 - x_1)$, откуда

$k=\frac{y_2-y_1}{x_2-x_1}, k(y_2+y_1)=\left(x_2^2+x_1x_2+x_1^2\right)+a$, аналогично,

$k=\frac{y-y_1}{x-x_1}, k(y+y_1)=\left(x^2+x_1x+x_1^2\right)+a$, и последняя формула

$k=\frac{y-y_2}{x-x_2}, k(y+y_2)=\left(x^2+x_2x+x_2^2\right)+a$

Вычтем из третьей формулы вторую,
получим $k(y_2-y_1)=x(x_2-x_1)+(x_2-x_1)(x_2+x_1)$
Откуда $k^2=x+x_2+x_1 \Leftrightarrow x=k^2-x_2-x_1$. Для координаты $y=y_1+k(x-x_1)=y_1+k\left(k^2-2x_1-x_2\right)$. Остаётся вспомнить, что для групповой операции нужно выбрать зеркальную
точку $(x,-y)=\left(k^2-x_2-x_1,-y_1+k\left(-k^2+2x_1+x_2\right)\right)$. (5)
Формула 3) доказана.
В случае перехода секущей в касательную получим $x_2=x_1, x=k^2-2x_1$
Далее дифференцируем уравнение 1) по x:

$$2yy'=3x^2+a, \Leftrightarrow k=y'=\frac{3x^2+a}{2y}=\frac{3x_1^{\ 2}+a}{2y_1},$$

Из формулы (5) получим $(x,-y)=\left(k^2-2x_1,-y_1+k\left(-k^2+3x_1\right)\right), k=\frac{3x_1^{\ 2}+a}{2y_1}$

Таким образом, доказана формула (4).
Циклическую группу образуют из множества точек эллиптической кривой (уравнение 1)), связанных геометрической групповой структурой (формулы 3),4)), дополняют полевой целочисленной структурой по модулю простого числа p, т.е. вместо 1) решают сравнения

$$y^2=x^3+ax+b(\mathrm{mod}\ p) \quad (6)$$

В конечном итоге мы пользуемся формулами 3)(либо 4)) и 6)., получая последовательно все точки э.к. циклической абелевой группы.

Напомним правила сравнений, обозначим целые числа $\overline{x},\overline{y},\overline{x_1},\overline{y_1}$ (всего возможно $p-1$ различных остатков по $\mathrm{mod}\ p$) $\overline{x_1}=\overline{x}\,\mathrm{mod}\ p, \overline{y_1}=\overline{y}\,\mathrm{mod}\ p$:

1) $\left(\overline{x}\pm\overline{y}\right)\mathrm{mod}\ p\equiv\left(\overline{x_1}\pm\overline{y_1}\right)\mathrm{mod}\ p$

2) $\left(\overline{x}\,\overline{y}\right)\mathrm{mod}\ p\equiv\left(\overline{x_1}\,\overline{y_1}\right)\mathrm{mod}\ p$ (7)

3) $\left(\frac{\overline{x}}{\overline{y}}\right)\mathrm{mod}\ p\equiv\left(\overline{x_1}\,\overline{y_2}\right)\mathrm{mod}\ p : \overline{y_2}*\overline{y_1}\equiv 1\mathrm{mod}\ p$ В этом случае элементы $\overline{y_2}=\overline{y_1}^{-1}, \overline{y_1}$ называются взаимно обратными по аксиоматике сравнений чисел по модулю p.

Как видно из формул 3) и 4) координаты точек э.к. являются рациональными числами, если первые 2 точки кривой также рациональные, т.е. геометрическая групповая операция оставляет

координаты точек рациональными и дальше. Правила (7) сужают множество рациональных точек, как правило, до конечного множества точек э.к. с целыми координатами x, y. Анализ формул (3) и (4) показывает, что если угловой коэффициент прямой принимает целочисленные значения в смысле формулы(7), то координаты x, y будут и дальше целыми. Таким образом, необходимо решить сравнение

$$\begin{cases} \left(\frac{y_2 - y_1}{x_2 - x_1}\right) \equiv (y_2 - y_1) \bmod p * (x_2 - x_1)^{-1} \bmod p, (x_2 - x_1)(x_2 - x_1)^{-1} \equiv 1 \bmod p \\ \frac{3x_1^{\,2} + a}{2y_1} \equiv (3x_1^{\,2} + a) \bmod p * (2y_1)^{-1} \bmod p, (2y_1) * (2y_1)^{-1} \equiv 1 \bmod p \end{cases} \quad (8)$$

Приведём пример поиска обратного элемента при решении сравнений, вызывающий наибольшие затруднения на практике.

$x^{-1} = \frac{1}{x} \bmod 11$: *если* $x \equiv 1 \bmod 11, x^{-1} = 1 \Leftrightarrow 1 * 1 \equiv 1 \bmod 11$

если $x \equiv 2 \bmod 11, x^{-1} = 6 \Leftrightarrow 2 * 6 = 12 \equiv 1 \bmod 11$

если $x \equiv 3 \bmod 11, x^{-1} = 4 \Leftrightarrow 3 * 4 = 12 \equiv 1 \bmod 11$

если $x \equiv 4 \bmod 11, x^{-1} = 3 \Leftrightarrow 4 * 3 = 12 \equiv 1 \bmod 11$

если $x \equiv 5 \bmod 11, x^{-1} = 9 \Leftrightarrow 5 * 9 = 45 \equiv 1 \bmod 11$

если $x \equiv 6 \bmod 11, x^{-1} = 2 \Leftrightarrow 6 * 2 = 12 \equiv 1 \bmod 11$

если $x \equiv 7 \bmod 11, x^{-1} = 8 \Leftrightarrow 7 * 8 = 56 \equiv 1 \bmod 11$

если $x \equiv 8 \bmod 11, x^{-1} = 7 \Leftrightarrow 8 * 7 = 56 \equiv 1 \bmod 11$

если $x \equiv 9 \bmod 11, x^{-1} = 5 \Leftrightarrow 9 * 5 = 45 \equiv 1 \bmod 11$

если $x \equiv 10 \bmod 11, x^{-1} = 10 \Leftrightarrow 10 * 10 = 100 \equiv 1 \bmod 11$

Краткое описание алгоритма построения последовательности точек.

1) Находим обратный элемент в (8) к $2y_1$ либо к $x_2 - x_1$.

2) Находим числа $k_1 = (y_2 - y_1) \bmod p * (x_2 - x_1)^{-1} \bmod p$ либо $k_1 = (3x_1^{\,2} + a) \bmod p * (2y_1)^{-1} \bmod p$

3) Находим числа

$$\begin{cases} x = (k_1^{\,2} - x_1 - x_2) \bmod p \\ y = (-y_1 + k_1(2x_1 + x_2 - k_1^{\,2})) \bmod p \end{cases} \quad (9)$$

Либо по формулам

$$\begin{cases} x = (k_1^{\,2} - 2x_1) \bmod p \\ y = (-y_1 + k_1(3x_1 - k_1^{\,2})) \bmod p \end{cases} \quad (10)$$

Координаты точки с использованием формул (8),(9),(10) дают новую точку э.к. с учётом всех отмеченных требований.

Первый основной этап алгоритма заключается в поиске порядка циклической абелевой группы, т.е. единицы по сложению - O и записи таблицы точек э.к., начиная с образующего элемента G. Как было сказано, элементы шифра с номером $k > n0$ находятся по формуле $P_k = P_{k-n0*s}\ 1 \le k - n0 * s \le n0 - 1, s \in N$.

Описание алгоритма шифрования и дешифрования

1) Все ключи, каждый символ передаваемого сообщения, порождающий элемент группы являются точками эллиптической прямой с целочисленными координатами.
2) Пусть абонент А передаёт абоненту В сообщение m,состоящее из последовательности символов. Каждый символ, переводится в число равное x-y, где x и y координаты некоторой точки группы точек э.к. При этом длина символьной строки выбирается 80 символов. Это делается из следующих соображений. Во – первых, разность $x - y$ пробегает подряд не весь ряд целочисленных значений, начиная с нуля до числа равному размерности алфавита. Во - вторых, имея длинную кодовую строку ключ можно увеличить пространство ключей до такой степени, что их перебор даже при известных параметрах кривой a, b, p и образующего элемента группы G, становится невозможным для злоумышленника. Даже для алфавита из 36 символов - 26 букв английского алфавита и 10 цифр, минимальная кодовая строка имеет число переборов $36! \approx 3.7 * 10^{41}$. Пусть суперкомпьютер может анализировать шифры со скоростью миллиард шифров в секунду. При этом понадобится времени порядка $3.7 * 10^{32} c = 10^{25} лет$. Что реально невозможно для разгадывания кодовой строки методом перебора даже при известных параметрах эллиптической кривой и образующем элементе группы G.
3) Абонент А выбирает случайное число $k < p$. Абоненту А известен открытый ключ абонента А- точка эллиптической кривой с координатами $P_b(x_b, y_b) = n_b G$.Под шифром эллиптической кривой понимают обе координаты 2 точек эллиптической кривой (всего 4 целых числа для каждого символа передаваемого сообщения), построенных по следующему правилу:

Символ алфавита S_k переводится в число $m_k = x_k - y_k$, для которого находится некоторая точка э.к $P_m(x_k, y_k)$., разность координат которой $x_k - y_k$ даёт данное целое число m_k из списка символьной строки алфавита (точек э.к. может быть несколько из группы точек э.к. - за этот счёт можно дополнительно усложнить зашифрованное сообщение).

4) $shifr = (kG, kP_b + P_m)$.

Отметим, что обе точки шифра обладают требованиями конфиденциальности. Во – первых, крипто аналитик по первой точке не сможет определить образующий элемент группы G, так как он имеет другую точку kG. С другой стороны, злоумышленник не сможет получить сообщение P_m, так как оно не равно точке $kP_b + P_m$.

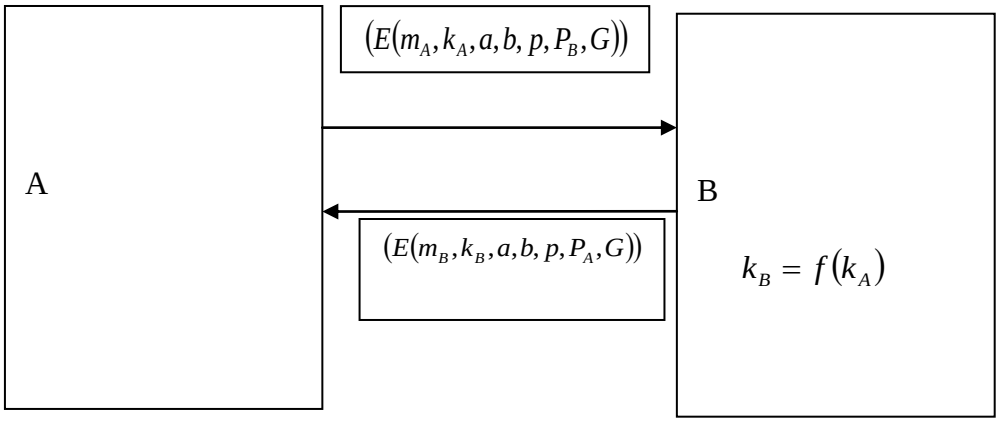
5) Тогда абонент В, зная свой секретный шифр n_b дешифрует данный зашифрованный символ по формуле

$$P_m = kP_b + P_m - n_b * kG = kn_bG + P_m - n_b * kG = P_m$$

В силу перестановки элементов групповой операции в циклической абелевой группе.

Структурная схема протокола обмена данными

Пусть два абонента договорились обмениваться данными с помощью точек эллиптической кривой с параметрами a, b и характеристикой поля p. Абонент А,зная открытый ключ В P_B, выбирает случайное число k_A и шифрует сообщение m_A методом эллиптической криптографии $E(m_A, k_A, a, b, p, P_B, G)$. Сообщение m_A и случайное число k_A и образующий элемент группы G в открытом виде не передаются, так как координаты шифра $(k_AG, k_A * P_A + m_A)$ их не содержат.



Абонент В дешифрует сообщение и от случайного числа k_A берёт хеш – функцию по формуле $k_B = f(k_A) \equiv k_{\max} \underset{k_{\max}=\max\{k_A,\overline{k_A}\},k_{\min}=\min\{k_A,\overline{k_A}\}}{\bmod} k_{\min}$, где $\overline{k_A}$ - число k_A, прочитанное справа налево. Действительно, функция $f(k_A)$ является хеш – функцией, так как для, например, симметричных чисел типа 11, 121,.. она равна нулю, т.е. по k_B невозможно восстановить k_A и расшифровать по шифру $E(m_A, k_A, a, b, p, P_B, G)$ сообщение m_A, не зная k_A. Абонент В поступает аналогичным образом. В хеширует случайное число $k_B = f(k_A)$ и шифрует сообщение m_B обратно для А $E(m_B, k_B, a, b, p, P_A, G)$.

Сообщение m_B и образующий элемент группы G в открытом виде не передаются, так как координаты шифра ($k_B G$, $k_B * P_A + m_B$) их не содержат. Абонент А дешифрует сообщение, так как он знает параметры эллиптической кривой, свой закрытый ключ n_A, своё начальное случайное число k_A и хеш - функцию $k_B = f(k_A)$. Случайное число k_B по каналу связи не передаётся. Поэтому злоумышленник, зная код $E(m_B, k_B, a, b, p, P_A, G)$ и даже все параметры эллиптической кривой a, b, p, P_A, G – открытый ключ и образующий элемент группы, не сможет восстановить m_B без случайного числа k_B. Отметим, что данная структурная схема является несимметричной.

Описание работы основной программы

1) Для загрузки основной программы необходимо ввести параметры эллиптической кривой - a, d, p, открытый ключ, являющийся целочисленной точкой эллиптической кривой (kx, ky), случайное число k, шифр - текст, состоящий из nn символов. Шифр - текст записывается как символьная переменная в массив $str[80]$. Далее, массив символов текста сравнивается с символами алфавитной строки, запоминается в массив $\text{int } kk[\text{int } nn]$ номер позиции соответствующего символа в алфавите.

2) Затем, используя образующий элемент группы $G = (\text{res}[0][0], \text{res}[1][0])$, заполняется массив $\text{res}[2][\text{N}]$ координатами целочисленных точек эллиптической кривой, образующих конечнопорождённую группу по формулам (1)-(10) первой главы. Число N согласно теории имеет оценку $N \le p + 2\sqrt{p} + 1$, где $p -$ характеристика поля конечнопорождённой группы точек э.к. Все координаты точек э.к. положительны, иначе, прибавляем к ним характеристику поля p.

Среди списка точек э.к. особое место занимает нейтральный элемент группы с номером в списке nol. Все точки списка э.к. массива $\text{res}[2][\text{N}]$ периодически повторяются, т.е.
$\left(\text{res}[0][\text{j}], \text{res}[1][\text{j}] = \text{res}[0][\text{nol} + \text{j}], \text{res}[1][\text{njl} + \text{j}], \text{j} = \overline{1, N - \text{nol}}\right)$.

3)Далее заполняется массив $xy[nol] = x - y$ как разность координат абсциссы и ординаты точек э.к. Все значения массива также должны быть неотрицательны.

4)На следующем этапе каждому элементу массива $kk[nn]$ сопоставляется некоторая точка из списка э.к. если

$kk[j] = xy[i], j = \overline{0, nn}, i = \overline{0, nol - 1}$. При этом заполняется массив $pm[2][nn]$, т.е. массив точек э.к. соответствующих строке исходного текста.

5)Выделяем элемент kG из массива точек э.к. $\text{res}[2][\text{N}], ((\text{res}[0][\text{k} - 1], \text{res}[1][\text{k} - 1]))$ которые служат первыми двумя координатами шифр - текста для массива $\text{shifr}[\text{j}][0], \text{shifr}[\text{j}][1], \text{j} = \overline{0, \text{nn} - 1}$.

6)В массив $\text{res1}[2][\text{N}]$ получают координаты точки $k * Pb$ последовательным сложением элемента Pb (точка $k * Pb$ имеет координаты $(\text{res1}[0][\text{k0} - 1], \text{k0}, \text{res1}[1][\text{k0} - 1])$).

7)Сложением точек массива $pm[2][nn]$ с элементом $k * Pb$ из массива $\text{res1}[2][\text{N}]$ заполняют массив $\text{shifr}[\text{j}][2], \text{shifr}[\text{j}][3], \text{j} = \overline{0, \text{nn} - 1}$ $k * Pb + pm$

Таким образом, мы получаем 4 целых числа для каждого исходного символа – координаты 2 точек эллиптической кривой

$k * G, k * Pb + pm = (\text{shifr}[\text{j}][0], \text{shifr}[\text{j}][1]), (\text{shifr}[\text{j}][1], \text{shifr}[\text{j}][2]), \text{j} = \overline{0, \text{nn} - 1}$.

8)Для дешифрования сообщения необходимо получить точку $n_b * kG$, для чего образуем массив $\text{de}[0][\text{n}_\text{b}], \text{de}[1][\text{n}_\text{b}]$, в котором точка $n_b * kG$ занимает позицию

$\text{de}[0][\text{n}_\text{b} - 1], \ \text{de}[1][\text{n}_\text{b} - 1]$

9)Для дешифрования сообщения образуют обратный элемент к точке $n_b * kG$ с координатами $-n_b * kG = (\text{de}[0][\text{n}_\text{b} - 1], -\text{de}[1][\text{n}_\text{b} - 1])$

10)Дешифрование есть сложение 2 точек эллиптической кривой $k * Pb + pm$ - последние две координаты массива $\text{shifr}[\text{j}][2], \text{shifr}[\text{j}][3], \text{j} = \overline{0, \text{nn} - 1}$ с точкой $-n_b * kG$ с координатами $\text{de}[0][\text{n}_\text{b} - 1], -\text{de}[1][\text{n}_\text{b} - 1]$, полученные по 2 первым координатам массива $\text{shifr}[\text{j}][0], \text{shifr}[\text{j}][1]$.

$k * Pb + pm - n_b * kG = k * n_b * G + pm - n_b * kG = pm$, или используя подпрограммы основной программы

$\text{xp}(\text{x1}, \text{y1}, \text{x2}, \text{y2}, \text{a}, \text{p}), \text{yp}(\text{x1}, \text{y1}, \text{x2}, \text{y2}, \text{a}, \text{p})$, получим:

$$(x_{pm}, y_{pm}) = \begin{pmatrix} xp(\text{de}[0][\text{n}_\text{b} - 1], -\text{de}[1][\text{n}_\text{b} - 1], \text{shifr}[\text{j}][2], \text{shifr}[\text{j}][3], \text{a}, \text{p}), \\ yp(\text{de}[0][\text{n}_\text{b} - 1], -\text{de}[1][\text{n}_\text{b} - 1], \text{shifr}[\text{j}][2], \text{shifr}[\text{j}][3], \text{a}, \text{p}) \end{pmatrix}, j = \overline{0, nn - 1}$$

11)Координаты дешифрования записываются в массив $(\text{otv}[0][\text{j}], \text{otv}[1][\text{j}]) = (x_{pm}, y_{pm}), \text{j} = \overline{0, \text{nn} - 1}$

12) Образуем массив данных $\text{des}[\text{j}] = x_{pm} - y_{pm}, \text{j} = \overline{0, \text{nn} - 1}$

13)Сравниваем значения массива с номерами соответствующих символов алфавитной строки и выводим символы расшифрованного текста

```
for(j=0;j<=nn-1;j++)
{
jj=des[j];
```

```
printf("%c \n",str[jj]);
}
```

14) Основная программа образует два текстовых файла *balka*1.*txt* и *balka*2.*txt* . В *balka*2.*txt* записывается случайное число k .Запись случайного числа нужна только для полноты протокола администратора При дешифровании знание отдельно взятого числа k не нужно. В текстовом файле *balka*1.*txt* содержится сам шифр. Если изменить несколько цифр в строках данного файла, то при дешифровании будут неправильно распознаны те символы, которым соответствуют строки в записи *balka*1.*txt* .

15)В процессе шифрования и дешифровании текста методом эллиптических кривых в конце операции составляется протокол шифрования и дешифрования и правильность проведения операции по каждому отдельному символу исходного текста с помощью вспомогательной функции $\text{int prov(int x1, int y1,int a, int b, int p)}$, т.е. проверяется принадлежность точки с координатами (x_1, y_1) эллиптической кривой с параметрами a, b в конечном поле целых чисел с характеристикой p .

16) В процессе шифрования и дешифрования данных необходимо использовать одни и те же параметры $a, b, p, k, (kx, ky)$, иначе, процесс дешифрования не осуществляется верно. Эту программу можно положить в основу создания генератора эллиптических кривых.
Код основной программы содержится в приложении.

17) Кроме того, в программном пакете содержится упрощённая основная программа, выполняющая только функцию дешифрования по исходным текстовым файлам. Она считывает текст $(\text{shifr[j][0], shifr[j][1]}), (\text{shifr[j][1], shifr[j][2]}), \text{j} = \overline{0, \text{nn - 1}}$ из файла *balka*1.*txt* и возвращает по этим данным исходный символьный текст m длиной nn символов.

Описание работы вспомогательных программ

1) *int nu(int t)* - определяет число разрядов целого числа в десятичной системе (отображает целое число в целое число $\text{int } t \to nu(\text{int } t)$).

2) *int xyz(char cx)* - вызывает символьную строку из текстового файла и преобразует посимвольно её в цифру десятичной системы счисления, в противном случае, если преобразованный символ не цифра, то конец работы с данной символьной строкой ($char\ \ cx \to \text{int}(char\, cx)$) .

3) *char zyx(int m, int nol)* - m, nol – входные целочисленные переменные программы, второе – номер нейтрального элемента конечнопорожденной

группы эллиптической кривой. Открывается файл *balka1,txt* . Данная подпрограмма заполняет файл *balka1,txt* шифр - текстом, последовательно принимая шифр *m* из основной программы (каждому символу исходного текста соответствует 4 числа – координаты и 4 строки в файле *balka1,txt*).

4) *int xp(int x1,int y1 ,int x2,int y2,int a , int p)* - по известным 2 целочисленным точкам, лежащим на эллиптической кривой $(x_1, y_1),(x_2, y_2)$, параметрам э.к. a, b, p получает x_3 - абсциссу третьей точки, принадлежащую э.к., в самом произвольном случае. Если точки $(x_1, y_1),(x_2, y_2)$ различны, то используются формулы сложения. Если точки $(x_1, y_1),(x_2, y_2)$ совпадают и $y_1 = y_2 \neq 0$, то используются формулы удвоения. Если $x_1 = x_2, y_1 = -y_2 \Leftrightarrow y_1 + y_2 \equiv 0 \bmod p \Leftrightarrow y_1 + y_2 \equiv p \bmod p$, то через исходные точки проходит вертикальная прямая, и следующая точка (x_3, y_3) является бесконечно удалённой или нейтральным элементом конечнопорождённой группы.

5) *int yp(int x1,int y1 ,int x2,int y2,int a , int p)* - аналогично функции *int xp(int x1,int y1 ,int x2,int y2,int a , int p)* образует ординату третьей точки э.к. по 2 известным точкам э.к. в произвольном случае.

6) *int prov(int x1,int y1,int a,int b,int p)* - позволяет проверить принадлежит ли точка с координатами (x_1, y_1) эллиптической кривой с параметрами a, b, p. Если точка принадлежит кривой, то функция $prov(x_1, y_1, a, b, p)$, запущенная из основной программы возвращает число 0. В противном случае $prov(x_1, y_1, a, b, p)$ возвращает другое число.

Число *nol* определяется по всему массиву целочисленных точек э.к. $nol : \underbrace{G + G + ... + G = O}_{nol}$.

Код вспомогательных программ содержится в приложении.

ШИФРОВАНИЕ, ДЕШИФРОВАНИЕ И ТЕСТИРОВАНИЕ

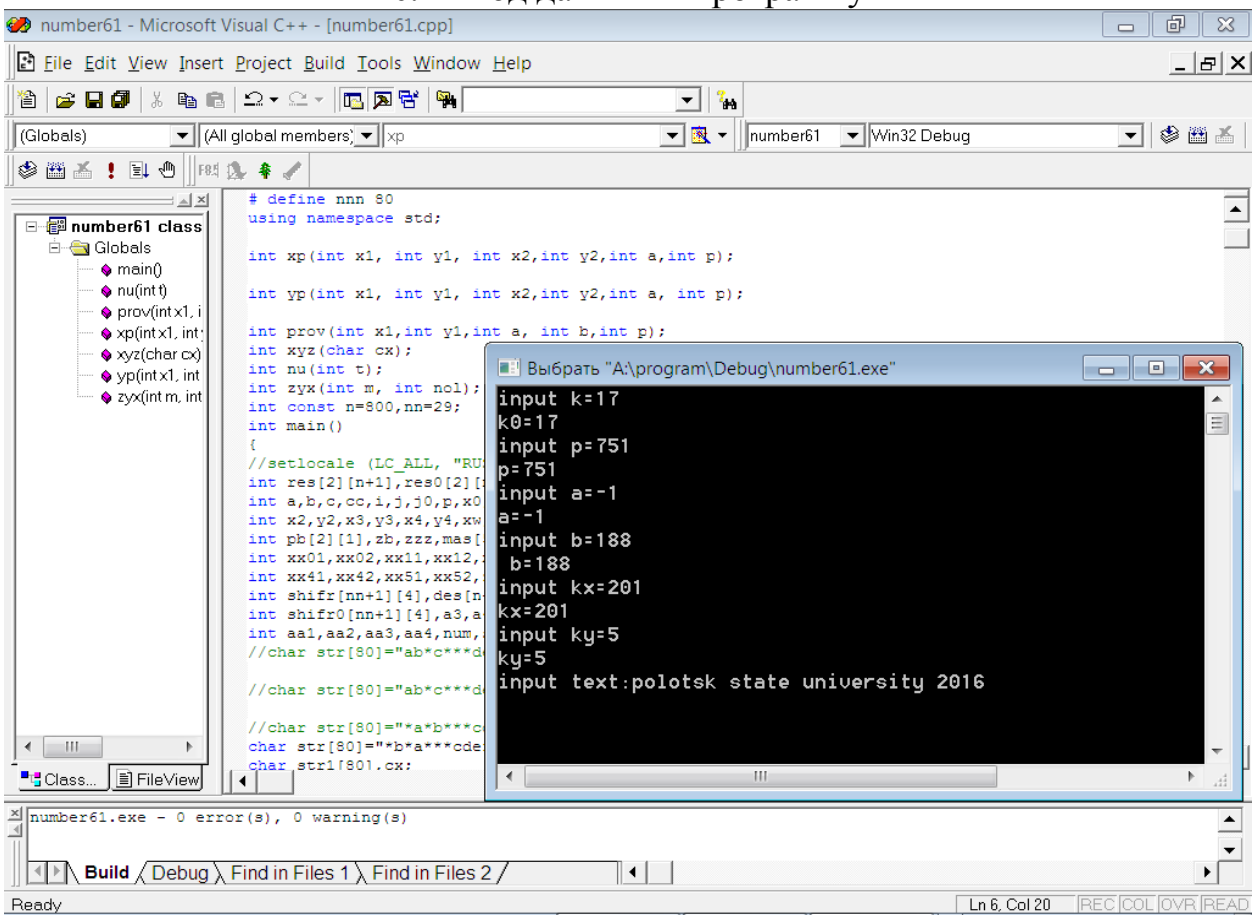
Примеры шифрования данных

Для тестирования шифрования данных введём фразу с латинским шрифтом Полоцкий государственный университет 2016(polotsk state university 2016) с длиной строки $nn = 29$. Результат ввода текста, параметры эллиптической кривой $a = -1, b = 188, p = 751$, открытый ключ $(kx, ky) = (201,5)$ приведены на рис1.

На рис.2 мы видим результат шифрования чёрным цветом. Каждому исходному символу текста соответствует четыре координаты 2 точек эллиптической кривой, расположенных в одной строке. Для удобства

ввода и построчного считывания тот же шифр в один столбец записывается в текстовый файл *balka1,txt* . Мы видим полное совпадение шифра на двух этапах.
Кроме того, видно также, что вводимое в программу случайное число $k = 17$ (рис.1) и записанное программой в текстовый файл *balka2.txt* также совпадают.

Рис.1 Ввод данных в программу



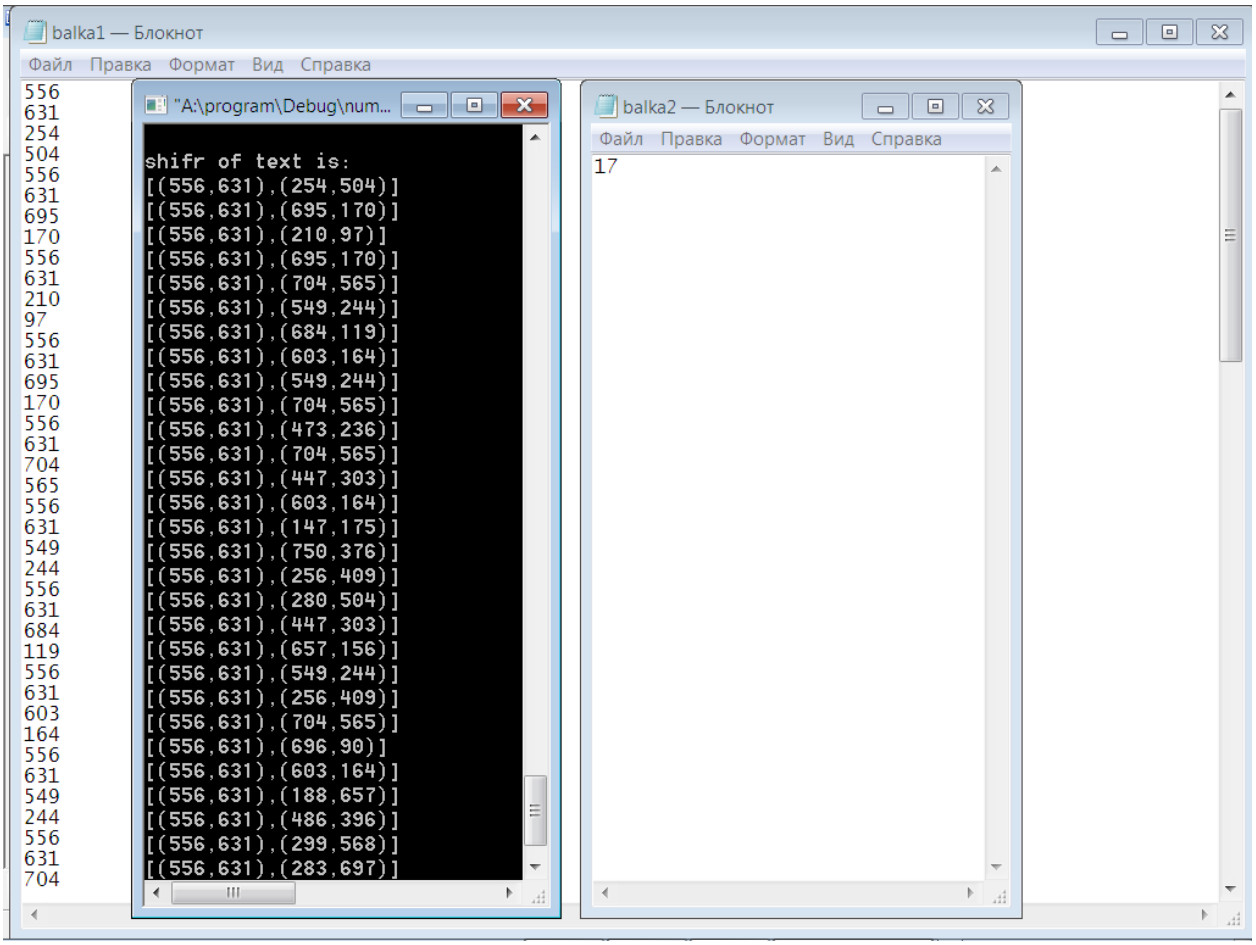


Рис 2. Сравнение программного шифра и шифра текстового файла.

В ходе работы программы составляется протокол шифрования по каждому шифруемому входному символу. Функция проверки $prov(x_1, y_1, a, b, p)$ осуществляет принадлежность каждой проверяемой точки с координатами (x_1, y_1) эллиптической кривой с параметрами a, b, p. Если точка принадлежит э.к. , то программа проверки возвращает число 0, в противном случае возвращается другое целое число. В протоколе шифрования указываются координаты точки эллиптической кривой, соответствующей каждому входному символу. Разность координат точки $x - y$ равняется позиции исходного символа в алфавитной стоке.

Например, по точке эллиптической кривой с координатами (xm, ym) мы можем определить $(xm, ym) = (680,657), des = xm - xy = 680 - 657 = 23$ исходный символ, используя алфавитную строку (нумерация символов в алфавитной строке начинается с нуля, поэтому $des = 23$ соответствует 24 символу, т.е. латинской букве p).

char str[80]="*b*a***cdefghi*jkl**mnopqrs**tuvwxyz01*2**3456789** ";

Действительно, фраза(polotsk state university 2016) начинается с буквы p.

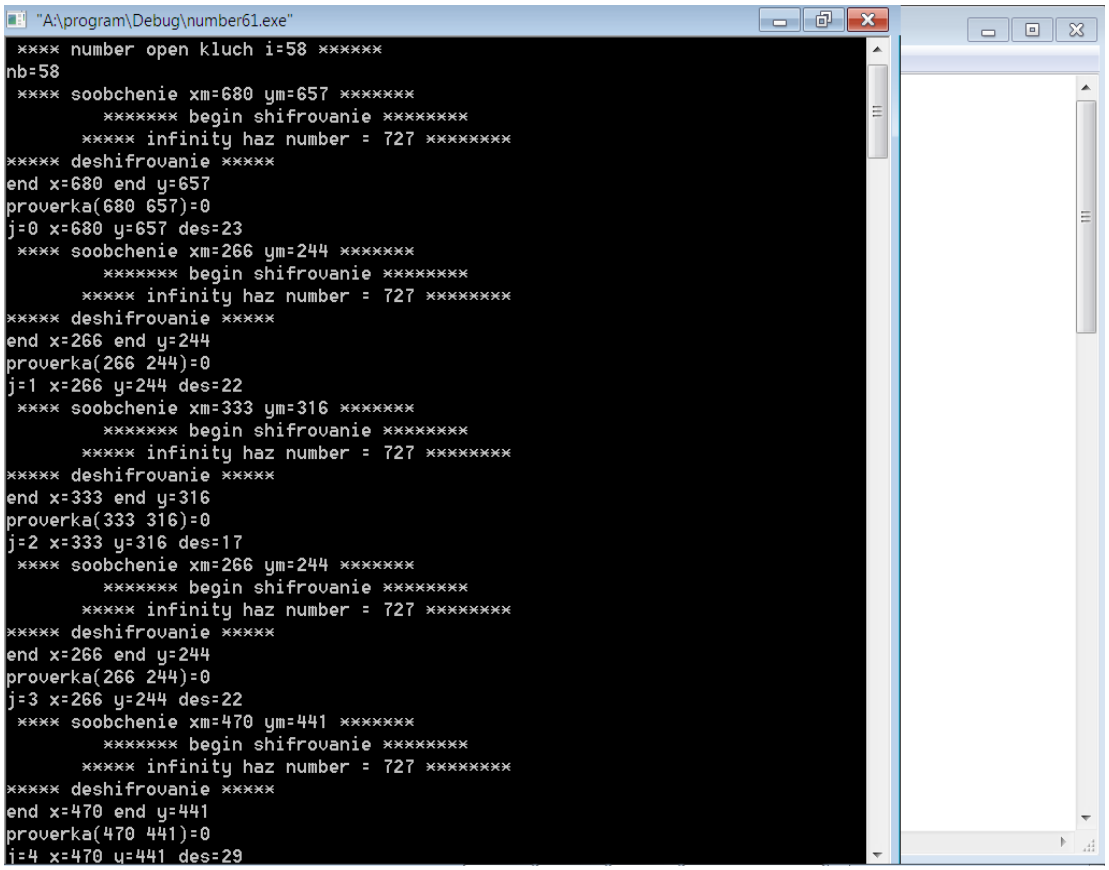


Рис 3. Протокол шифрования.

Второй символ $(xm, ym) = (266,244), des = xm - xy = 266 - 244 = 22$ соответствует 23 по счёту символу в алфавитной строке, т.е. латинской букве o, что соответствует второй букве в слове polotsk. Мы видим из рисунка 3, что все точки текста дают функцией проверки значение ноль, т.е. все точки являются точками эллиптической кривой.

3.2 Примеры дешифрования данных

Дешифрование данных возможно основной программой, считывающей шифр из текстового файла *balka1,txt* . Дешифрование осуществляется по формулам пунктов 10)-11и алгоритму12)-13). Мы видим, что исходная фраза рис.1 и конечная рис.4 (polotsk state university 2016) полностью совпадают, учитывая пробелы между словами.

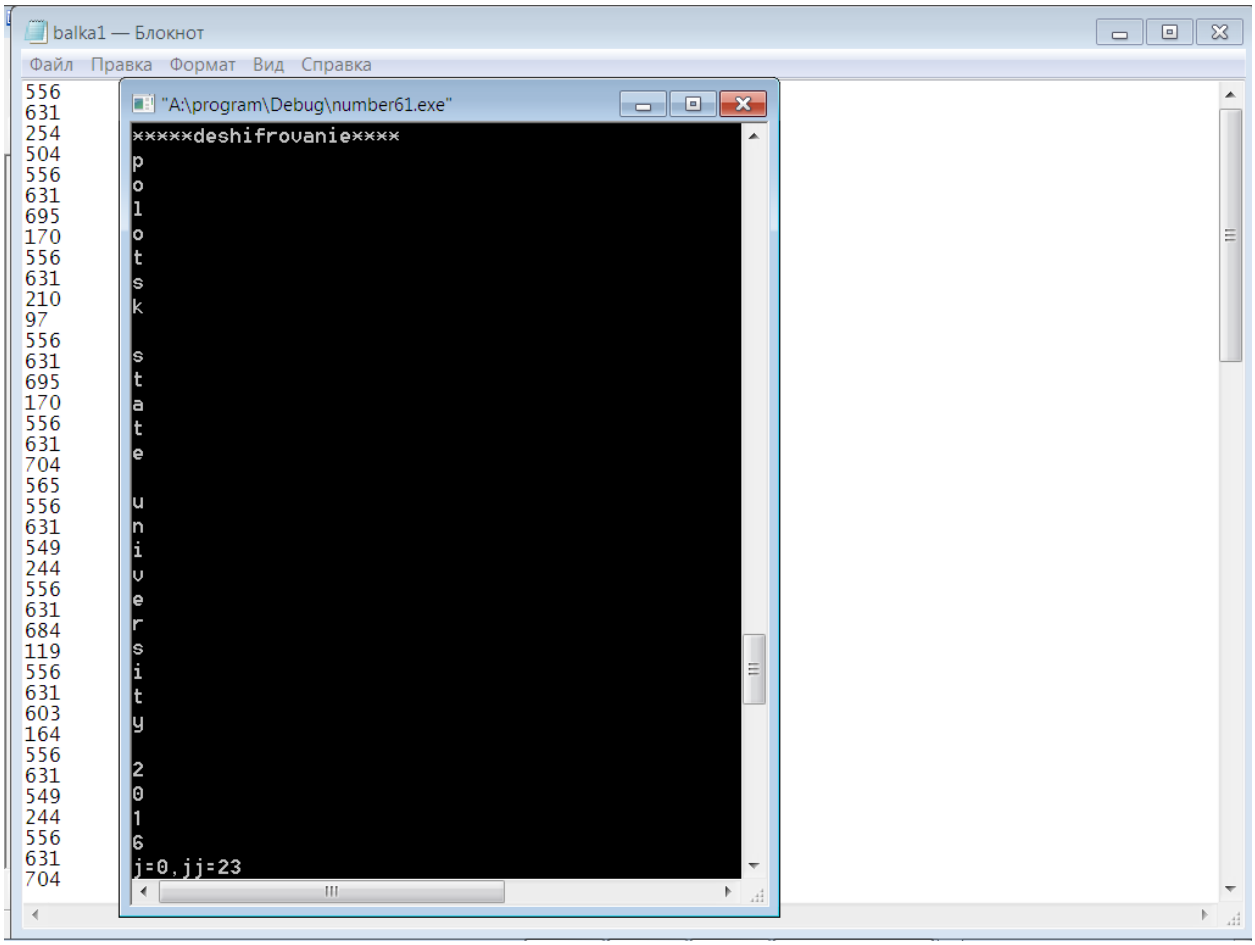


Рис 4. Дешифрование данных.

Дешифрование исходных данных возможно другой независимой программой, предназначенной только для работы с текстовым файлом *balka1,txt* . Функция дешифрования использует только параметры кривой $a=-1, b=188, p=751$, открытый ключ $(kx, ky)=(201,5)$ и шифр – текст из файла *balka1,txt* .

Рис.5 Ввод данных при дешифровании

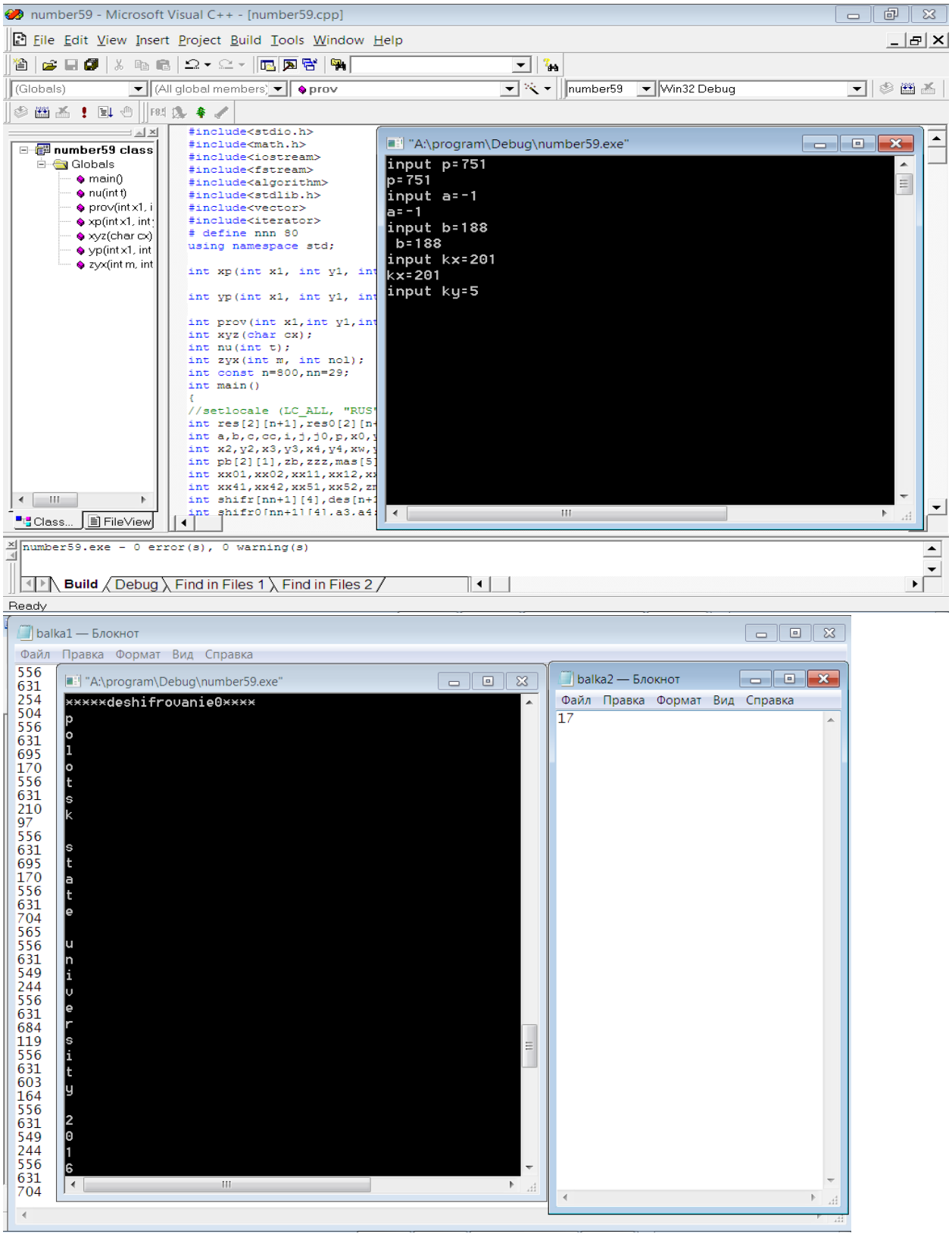


Рис.6 Дешифрование данных

Мы видим, что при дешифровании обеими программами из записанного в текстовом файле *balka1,txt* шифра эллиптической кривой, получается исходный текст рис.1 - polotsk state university 2016 .

Интерфейс пользователя

В качестве интерфейса пользователя программы ввода данных можно рассмотреть, например, следующий интерфейс с вводом длины сообщения в символах *nn* , параметры эллиптической кривой *a, b, p* , открытый ключ (kx, ky), случайное число *k* .

Интерфейс включает следующие окна – окна ввода чисел *a, b, p* , окна ввода открытого ключа *Open Key – X , Open Key – Y* , окно ввода случайного числа *Random number k* .

Функциональные элементы запуска, очистки и остановки интерфейса выхода из программы *Run, Clear , Stop, Out* .

Окно ввода первичной информации *Sourse text* , куда вводится исходный текст с помощью клавиатуры и установки курсора в окно ввода.

Окно шифрованного текста непосредственно перед записью шифра в текстовый файл 1.txt *Writing text*.

Длина записи исходного текста в окне *nn* не является активной, т.е. в это окно ничего не заносится. Длина строки вводимого текста *nn* автоматически изменяется в данном окне при удалении или добавлении каждого нового символа.

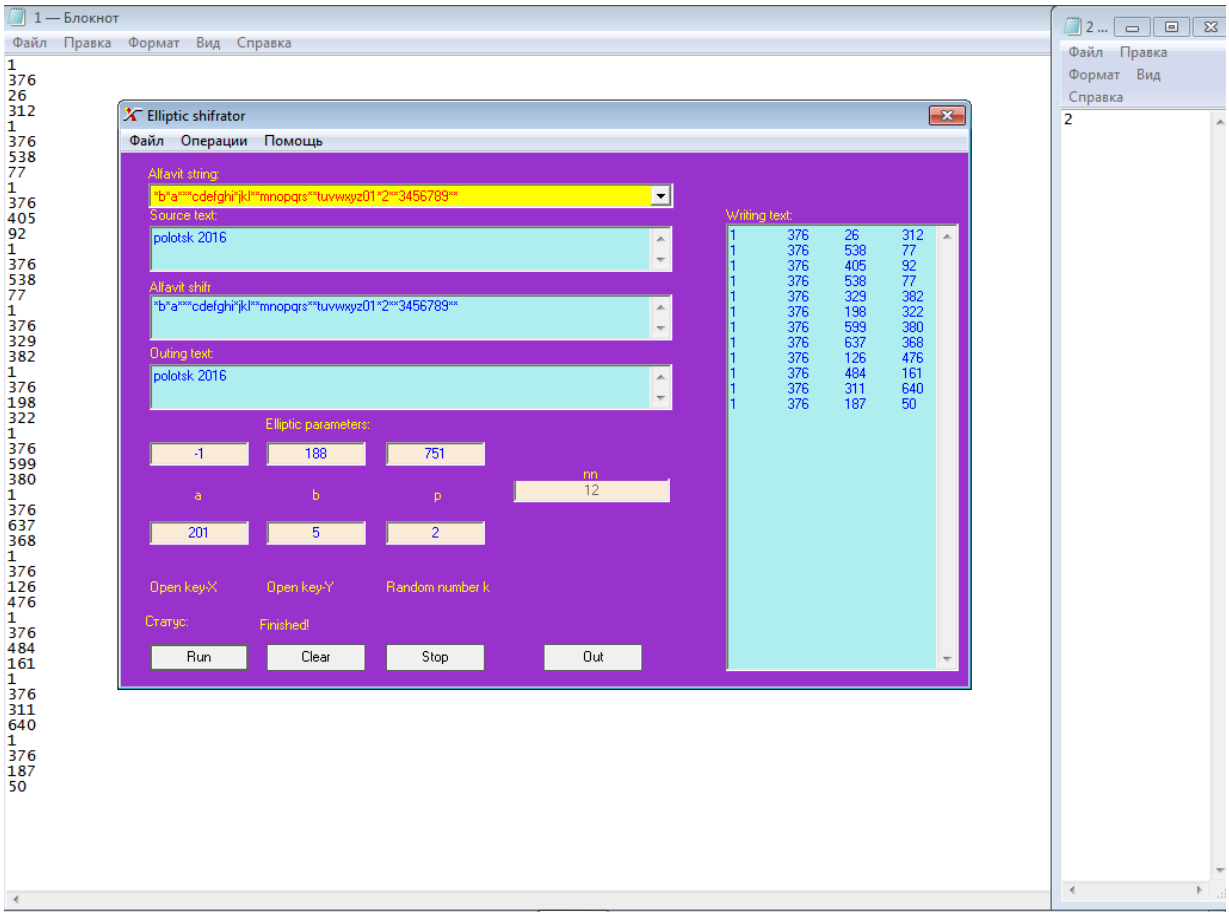


Рис.7 Шифрование данных интерфейсом пользователя

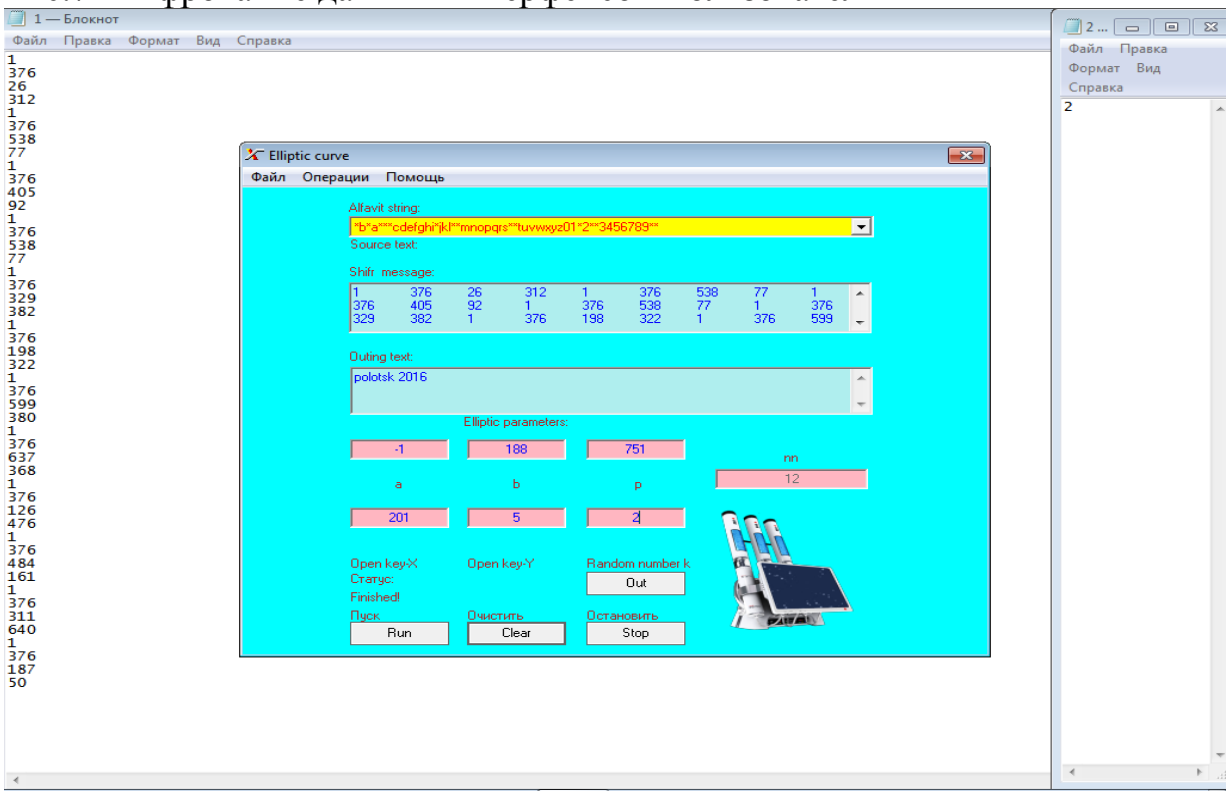


Рис.8 Дешифрование интерфейсом пользователя исходного текста.

Окно дешифрованного текста *Outing text* выводит текст, посимвольно совпадающий с исходным текстом.
Данный интерфейс не только шифрует исходный текст, но и является программой – контролем на всех этапах шифрования и дешифрования и создания текстового файла для шифра.
Интерфейс содержит также функциональные окна с падающим списком опций, с помощью которых также можно шифровать исходный текст (например, операции – кодировать, закрыть; файл – просмотреть; помощь – о программе).

Сравнивая рис.7 и рис.8 мы видим, что при запуске интерфейса параллельно происходит как шифрование исходного текста, так и создание текстового файла для записи шифра 1.txt. Длина входной фразы в символах равна 12 символов. На этапах шифрования (вертикальное окно интерфейса), записи шифра в текстовый файл (левый столбец на рис.8 переданного из файла 1.txt), полностью совпадают. Совпадают также посимвольно тексты исходный (в окне *Soursetext*) и дешифрованный текст(в окне *Outing text*).
Кроме того, интерфейс программы содержит пассивный логический оператор “Статус”, который в случае успешного запуска возвращает состояние *Finished* . В случае ошибки данный логический оператор указывает состояние(род произошедшей ошибки).
Справа на рис.8 указано случайное число k=2, записанное программой в текстовый файл 2.txt. Оно совпадает с числом, введённым в интерфейс в окно *Random number k* . В приложении D содержится модуль интерфейса программы.
Обратим внимание, что кодовые алфавитные строки на рис.7,рис.8 при шифровании и дешифровании текста должны совпадать.
*char str[80]="*b*a***cdefghi*jkl**mnopqrs**tuvwxyz01*2**3456789** ";*

Замечание:
Программы шифратора и дешифратора можно оформить в виде отдельных библиотек dll. Такие библиотеки легко подключить к интерфейсам, написанных на любых объектно - ориентированных языках, например, Compaq Visual Fortan, Visual C++ и других.

Интерфейс администратора

Интерфейс администратора в отличие от интерфейса пользователя имеет координаты образующего элемента конечно порождённой группы Gx, Gy. Это сделано для того, чтобы администратору было проще тестировать алфавитные строки – ключи под заданную эллиптическую кривую. По умолчанию выбраны параметры кривой
*a=0,b=-4,p=211,kx=115,ky=48,Gx=Dy=2,k=17,Alfavit string: str[80]="ab*c***d**ef*gh*i*j*kl*m**n**o**pqrs*t**u**v*wxyz012****3**4*56*789*** ";*
Шифруем фразу:*polotsk 2016 june 30*

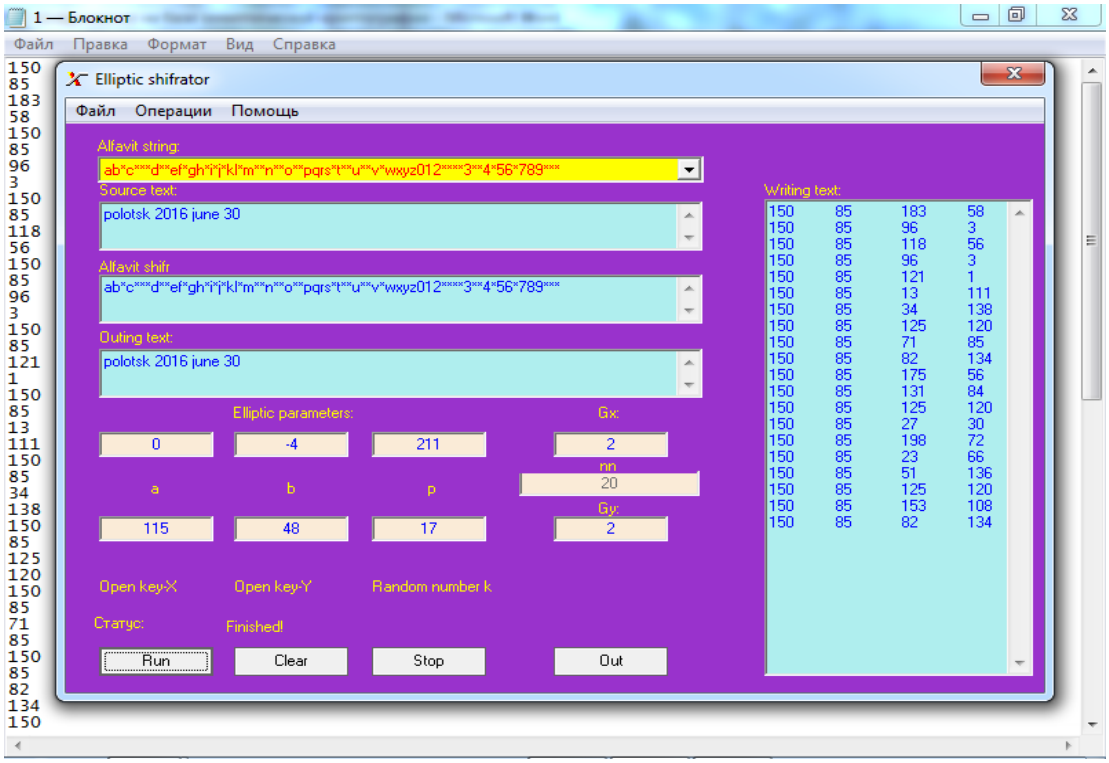


Рис.9 Шифрование интерфейсом администратора исходного текста.

Как видно из рис.9 контрольная строка *auting text* совпадает с со строкой исходного текста *sourse text*, а шифр текста в строке *writing text* совпадает с шифром, записанным в текстовый файл 1.txt.
Программа дешифратора по умолчанию настроена на эллиптическую кривую с параметрами

a=0,b=-4,p=211,kx=115,ky=48,Gx=Dy=2,k=17,Alfavit string:
*str[80]="ab*c***d**ef*gh*i*j*kl*m**n**o**pqrs*t**u**v*wxyz012****3**4*56*789*** ";*
Для дешифрования передаём файл 1.txt, в программу дешифратора (путь 2.26 - primer - debug) и запускаем программу дешифратора.

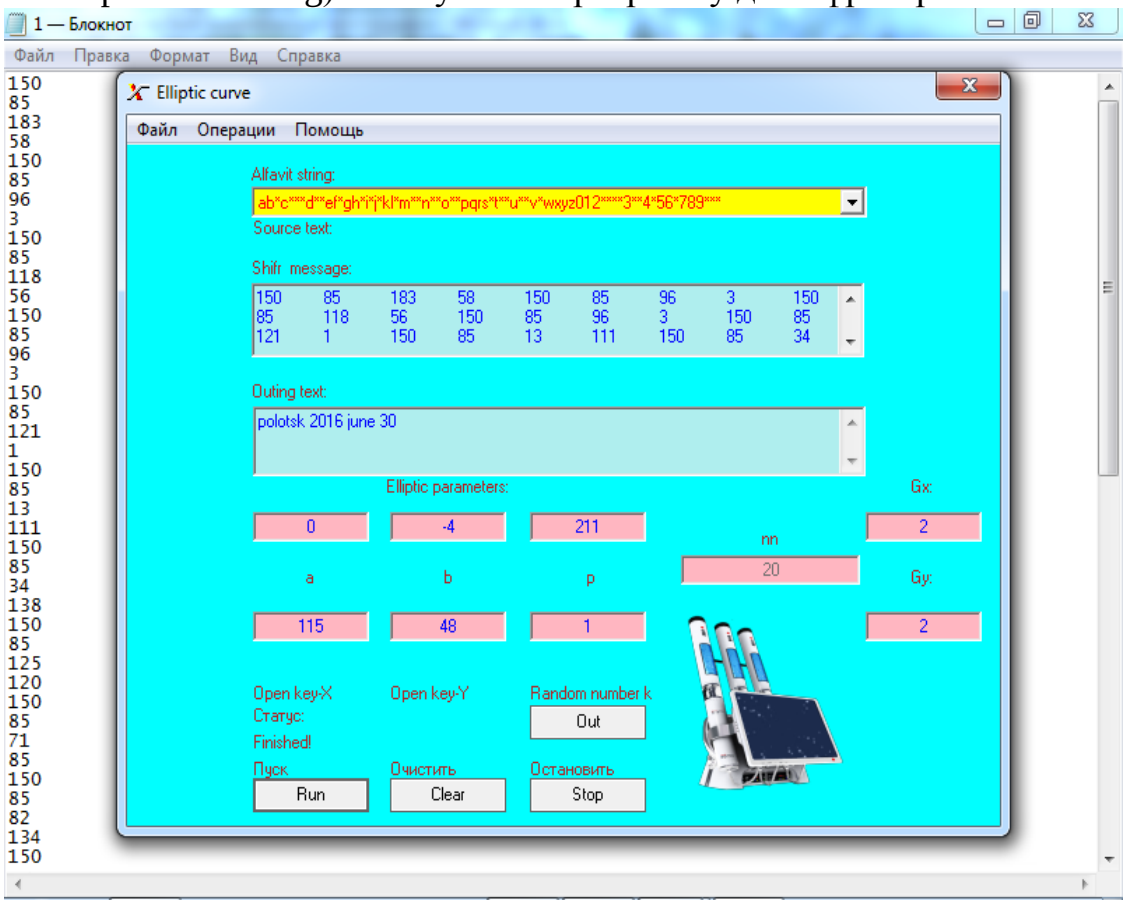


Рис.10 Дешифрование интерфейсом администратора исходного текста.

Из рис.10 видно, что шифр – текст, прочитанный из файла 1.txt совпадает с шифром прочитанной программой дешифратора и введённым в окно shifr message. Дешифрованное сообщение с теми же параметрами эллиптической кривой, что и у шифратора тождественно равно исходному сообщению *polotsk 2016 june 30.*

Рассмотрим теперь шифрование фразы *polotsk 2016 june 30* на базе эллиптической кривой с параметрами:
a=-1,b=188,p=751,kx=201,ky=5,Gx=0,Gy=376,k=11,Alfavit string:
*char str[80]="*b*a***cdefghi*jkl**mnopqrs**tuvwxyz01*2**3456789** ";*

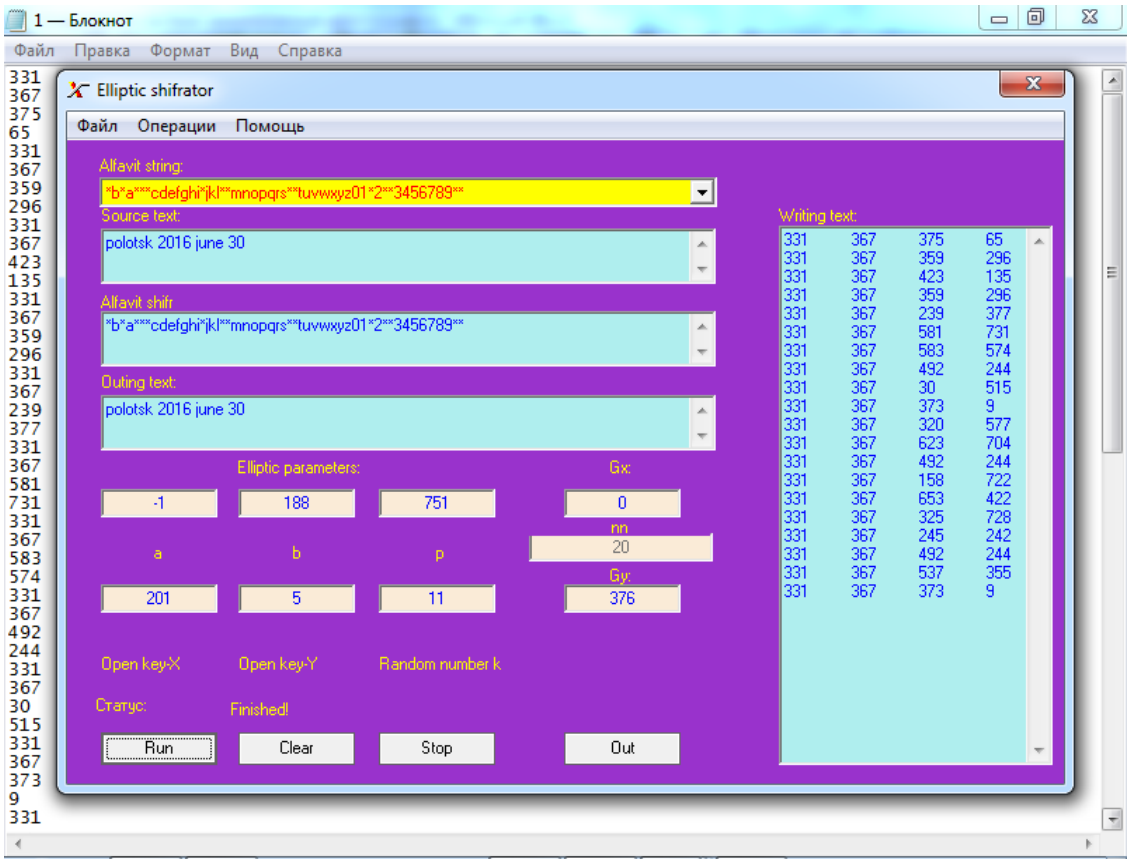


Рис.11 Шифрование интерфейсом администратора исходного текста.

Как видно из рис.11, контрольная строка *auting text* совпадает со строкой исходного текста *sourse text*. Шифр текста в строке *writing text* совпадает с шифром, записанным в текстовый файл 1.txt. В то же время шифр из файла 1.txt рис.11 отличается от шифра из файла 1.txt рис.9.

Дешифруем текст, пользуясь программой дешифратора, сгенерированным файлами 1.txt и параметрами эллиптической кривой: *a=-1,b=188,p=751,kx=201,ky=5,Gx=0,Dy=376,k=11,Alfavit string: char str[80]="*b*a***cdefghi*jkl**mnopqrs**tuvwxyz01*2**3456789** ";*

На рис.12 видно, что шифр – текст, прочитанный из файла 1.txt , совпадает с шифром прочитанным программой дешифратора и введённым в окно shifr message. Дешифрованное сообщение с теми же параметрами эллиптической кривой, что и у шифратора тождественно равно исходному сообщению *polotsk 2016 june 30*(рис.11).

Таким образом, интерфейсы шифратора и дешифратора для администратора позволяют синтезировать алфавитные строки – ключи как при фиксированных параметрах эллиптической кривой с фиксированной длиной алфавитной строки, так и получение новых алфавитных строк – ключей разной длины методом использования разных эллиптических кривых.

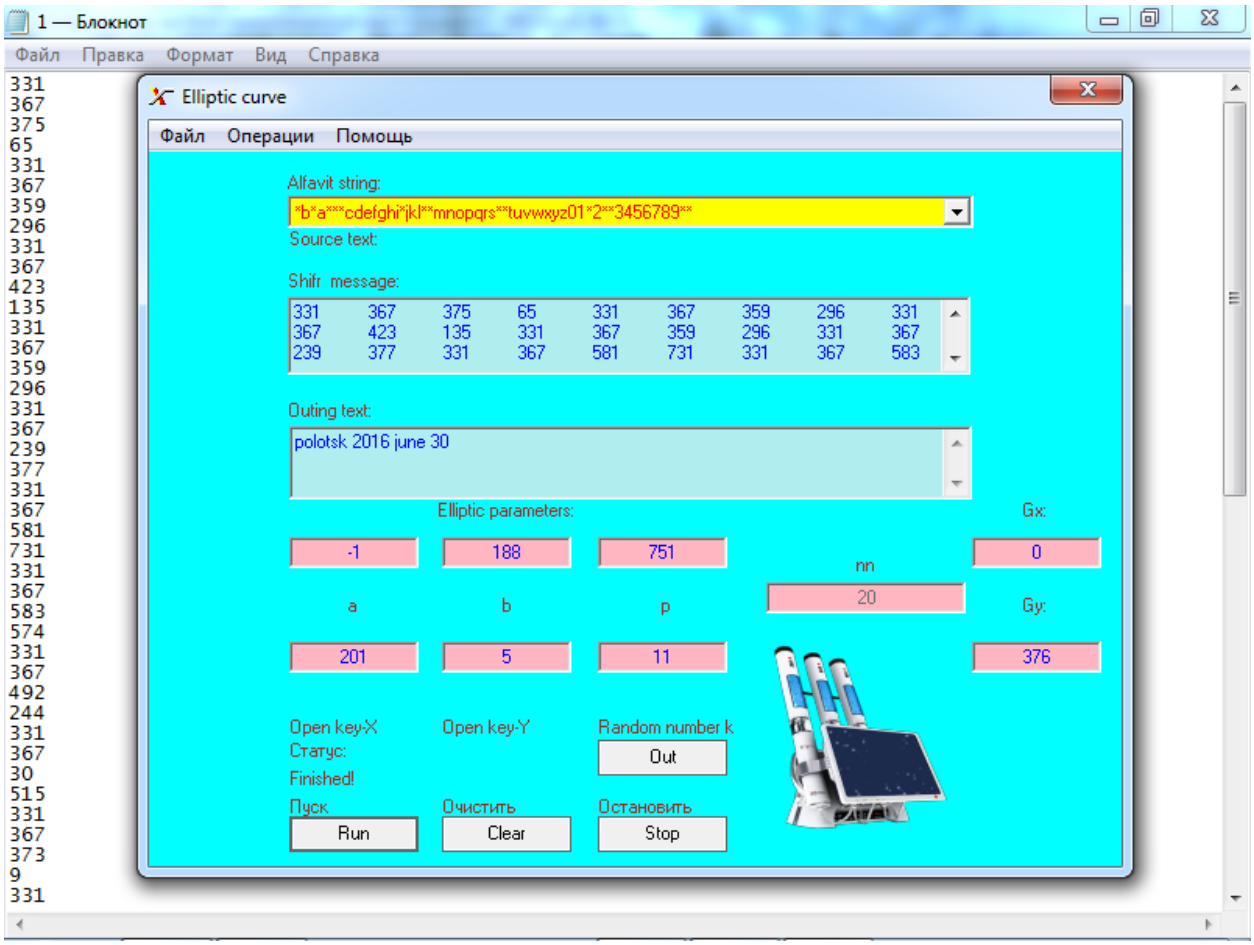


Рис.12 Дешифрование интерфейсом администратора исходного текста.

Интерфейсы администратора включают меню с использованием горячих клавиш файл (декодировать (alt+ctrl+D),выход(alt+C)), операции(очистить(alt+o),остановить(alt+s)), помощь(о программе(alt+a)). В результате чего запускать программу и некоторые её функции можно как из меню, так и с помощью горячих клавиш.

Литература:

1. Березин Б.В., Дорошкевич П.В. Цифровая подпись на основе традиционной криптографии: вып. 2.- М.:МП Ирбис – II,1992.-202 с.
2. Бутакова Н.Г., Семененко В.А., Федоров Н.В. Криптографическая защита информации: учебное пособие для вузов.- М.:Изд-во МГИУ, 2011. –т 316 с.
3. Жданов О.Н., Чалкин В.А. Эллиптические кривые: Основы теории и криптографические приложения.- М.: Книжный дом ЛИБРИКОМ, 2013.- 200с.
4. Жданов О.Н., Золотарёв В.В. Методы и средства криптографической защиты информации .167с.
5. Болотов А.А., Гашков С.Б., Фролов А.Б., Часовских А.А. Элементарное введение в эллиптическую криптографию.
6. Recjmmended Eliptic Curves for Govement Use.
7. SEC 2/ Recommended Eliptic Curves Domain Parametres
8. Schoof's algoritm.
9. Schoof – Elkies – Atkin algoritm .
10. Neal Koblitz. Random Curves: Journeys of a Mathematician, - Spinger, 2009.- C. 312 -313. – 402 с. – ISBN 9783540740780.
11. Koblitz N.A. Course in namber theory and cryptography. – USA: Spinger – Verlag?1994.- 235с.
12. Ишмухаметов Ш.Т. Методы факторизации натуральных чисел.- Казань:КФУ,2011.-1990 с.
13. ГОСТ Р 34.10- 2012.

ПРИЛОЖЕНИЕ 1

Вспомогательные программы

```
int nu(int t)
{
int i,t1,sss;
t1=t;
sss=0;
for(i=0;i<=100;i++)
{
if(t1>=10)
{
t1=(t1-t1%10)/10;
sss=sss+1;
}
else if(t1<10)
{
t1=t1;
sss=sss+1;
i=100;
}
```

```
}
return sss;
}
//////////////
int zyx(int m, int nol)
{
ofstream out("balka1.txt" , ios::app);
out<< m ;
out<<endl;
printf("\n");
return 1.0;
}
////////////////
int xyz(char cx)
{
int k,i,j,nx;
char str0[10]={'0','1','2','3','4','5','6','7','8','9'};
 for(k=0;k<=9;k++)
{
if(cx==str0[k] )
{
```

```
nx=k;
}
}
return nx;
}
int xp(int x1,int y1,int x2,int y2,int a, int p)
{
int s,s1,s2,s3,s4,ss,w,i;
if(!(x2==x1))
{
s=(y2-y1)%p;
ss=(x2-x1)%p;
if(s<0)
{
s=s+p;
}
if(ss<0)
{
ss=ss+p;
}
if(!ss==0)
```

```
{
for(i=1;i<=p-1;i++)
{
s1=(i*ss)%p;
if(s1==1 )
{
w=i;
}
}
}
else if(ss==0)
{
w=p;
}
s1=(w*s)%p;
s2=(s1*s1-x2-x1)%p;
if(s2<0)
{
s2=s2+p;
}
return s2;
```

```
}
else if(x2==x1 && y2==y1 && !y1==0)
{
s=(3*x1*x1+a)%p;
if(s<0)
{
s=s+p;
}
s1=(2*y1)%p;
if(s1<0)
{
s1=s1+p;
}
if(!s1==0)
{
for(i=1;i<=p-1;i++)
{
ss=(s1*i)%p;
if(ss==1)
{
w=i;
```

```
}
}
}
else if(s1==0)
{
w=p;
}
s2=(w*s)%p;
s3=(s2*s2-2*x1)%p;
if(s3<0)
{
s3=s3+p;
}
return s3;
}
}
int yp(int x1,int y1,int x2,int y2, int a, int p)
{
int s,s1,s2,s3,s4,s5,ss,sss,w,i;
if(x2==x1 && y2==y1 && !y1==0)
{
```

```
s=(3*x1*x1+a)%p;
s1=(2*y1)%p;
if(s<0)
{
s=s+p;
}
if(s1<0)
{
s1=s1+p;
}
if(!(s1==0))
{
for(i=1;i<=p-1;i++)
{
ss=(i*s1)%p;
if(ss==1)
{
w=i;
}
}
}
```

```
else if(s1==0)
{
w=p;
}
s2=(w*s)%p;
s3=(3*x1-s2*s2)%p;
s4=(s3*s2-y1)%p;
if(s4<0)
{
s4=s4+p;
}
return s4;
}
else if(!(x2==x1))
{
s=(y2-y1)%p;
ss=(x2-x1)%p;
if(s<0)
{
s=s+p;
}
```

```
if(ss<0)
{
ss=ss+p;
}
if(!(ss==0))
{
for(i=1;i<=p-1;i++)
{
s1=(i*ss)%p;
if(s1==1 )
{
w=i;
}
}
}
else if(ss==0)
{
w=p;
}
sss=(w*s)%p;
s2=(-sss*sss+x2+2*x1)%p;
```

```
s3=(s2*sss-y1)%p;
if(s3<0)
{
s3=s3+p;
}
return s3;
}
}
int prov(int x1,int y1,int a,int b,int p)
{
int s,s1,s2,s3,s4;
s1=(y1*y1)%p;
s2=(x1*x1)%p;
s3=(s2*x1)%p;
s4=s3+(a*x1+b)%p;
s=(s1-s4)%p;
if(s<0)
{
s=s+p;
}
return s;
```

```
}
```

Приложение 2

КОД ОСНОВНОЙ ПРОГРАММЫ И БИБЛИОТЕКИ

```
#include<stdio.h>
#include<math.h>
#include<iostream>
#include<fstream>
#include<algorithm>
#include<stdlib.h>
#include<vector>
#include<iterator>
# define nnn 80
using namespace std;

int xp(int x1, int y1, int x2,int y2,int a,int p);

int yp(int x1, int y1, int x2,int y2,int a, int p);

int prov(int x1,int y1,int a, int b,int p);
int xyz(char cx);
int nu(int t);
int zyx(int m, int nol);
int const n=800,nn=32;
int main()
{
//setlocale (lc_all, "rus");
int res[2][n+1],res0[2][n+1],res1[2][n+1],nol,l,xy[n+1];
int a,b,c,cc,i,j,j0,p,x0,y0,x,y,x1,y1,k,kkk,zz;
int x2,y2,x3,y3,x4,y4,xw,yw,k0,t,a1,a2;
int pb[2][1],zb,zzz,mas[5][2],jj,xx2,yy2;
int xx01,xx02,xx11,xx12,xx21,xx22,xx31,xx32;
int xx41,xx42,xx51,xx52,zm,zz0;
int  pm[2][nn+1],jjj[nn+1],kk[nn+1],de[2][n+1],n1;
int shifr[nn+1][4],des[n+1],nb,otv[2][nn+1];
int shifr0[nn+1][4],a3,a4;
int aa1,aa2,aa3,aa4,num,ss,p1,t1,k00;
```

```
        //char
        str[80]="ab*c***defghij*klm**nopqrst**uvwxyz012*3**456789*";

        //char
        str[80]="ab*c***defghij*klm**nopqrst**uvwxyz012*3**456789*** ";

        //char
        str[80]="*a*b***cdefghi*jkl**mnopqrs**tuvwxyz01*2**3456789*";
char str[80]="*b*a***cdefghi*jkl**mnopqrs**tuvwxyz01*2**3456789**  ";
        char str1[80],cx;
        char str00=('0','1','2','3','4','5','6','7','8','9');
        char str2[12][nn+1];
        char str33[12][nn*4+1];
        int int2[12][nn+1];
        printf("input k=");
        scanf(" %d", &k0);
        printf("k0=%d \n",k0);
        printf("input p=");
        scanf("%d", &p);
        printf("p=%d\n",p);
        printf("input a=");
        scanf("%d", &a);
        printf("a=%d \n",a);
        printf("input b=");
        scanf("%d", &b);
        printf(" b=%d\n",b);
        printf("input kx=");
        scanf(" %d", &pb[0][0]);
        printf("kx=%d\n",pb[0][0]);
        printf("input ky=");
        scanf(" %d", &pb[1][0]);
        printf("ky=%d\n",pb[1][0]);
        printf("input text:");
        for(i=0;i<=nn;i++)
        {
        scanf("%c", &str1[i-1]);
        }
```

```
printf("proverka\n");
for (i=0;i<=nn-1;i++)
{
        printf("i=%d %c\n",i,str1[i]);
}
for(i=0;i<=nn-1;i++)
for(j=0;j<=80;j++)
{
        if(str[j]==str1[i])
        {
                kk[i]=j;
printf("i=%d kk=%d\n",i,kk[i]);
        }
}
p1=p;
ss=1;
for(k00=0;k00<=100;k00++)
{
p1=(p1-p1%10)/10;
if(p1>10)
{
ss=ss+1;
}
else
{
num=ss+1;
k00=100;
}
}
printf("num=%d\n",num);

res[0][0]=0;
res[1][0]=376;
x0=res[0][0];
y0=res[1][0];
x1=x0;
y1=y0;
```

```
//c=y1-p;
        xx2=xp(x0,y0,x0,y0,a,p);
        res[0][1]=xx2;
        yy2=yp(x0,y0,x0,y0,a,p);
        res[1][1]=yy2;
x2=xx2;
y2=yy2;
//verhnya ocenka m<=p+2*p^0.5+1
for(k=2;k<=p+sqrt(p)+1;k++)
        {
        xw=x2-x1;
if((xw==0) && (y1+y2==p))
{
        res[0][k]=-1000;
        res[1][k]=-1000;
        nol=k;
        printf("nol=%d\n",nol);
        res[0][k+1]=x0;
        res[1][k+1]=y0;
        res[0][k+2]=res[0][1];
        res[1][k+2]=res[1][1];
        x2=x0;
        y2=y0;
        x3=res[0][1];
        y3=res[1][1];
k=k+2;
}
else
{
        x3=xp(x1,y1,x2,y2,a,p);
        y3=yp(x1,y1,x2,y2,a,p);
res[0][k]=x3;
res[1][k]=y3;
}
printf("k=%d (%d %d)(%d %d)\n",k,x2,y2,x3,y3);
printf("prov(%d %d)=%d\n",x3,y3,prov(x3,y3,a,b,p));
x1=x0;
```

```
x2=x3;

y1=y0;
y2=y3;
}
printf("nol=%d\n",nol);
if(k0>nol)
{
k0=k0%nol;
}
for(i=0;i<=nol-1;i++)
{
x=res[0][i];
y=res[1][i];
l=(x-y);
if(l<0)
{
l=l+p;
}
xy[i]=l;
//printf("i=%d x-y=%d\n",i,xy[i]);
}
for(i=0;i<=nn-1;i++)
{
for(j=0;j<=nol-1;j++)
{
        if(xy[j]==kk[i] )
        {
                pm[0][i]=res[0][j];
                pm[1][i]=res[1][j];
                j=nol-1;
                }
                }
                printf("j=%d pmx=%d pmy=%d\n",i,pm[0][i],pm[1][i]);

}
for(i=0;i<=nn-1;i++)
```

```
{
        printf("i=%d prov(pmx=%d
pmy=%d)=%d\n",i,pm[0][i],pm[1][i],prov(pm[0][i],pm[1][i],a,b,p));
}

for(i=0;i<=nol-1;i++)
{
        res0[0][i]=res[0][i];
        res0[1][i]=res[1][i];
}
a1=res0[0][k0-1];
a2=res0[1][k0-1];
printf("kg=(%d %d) \n",a1,a2);
//return 0;
for(j=0;j<=nn-1;j++)
{
shifr[j][0]=a1;
shifr[j][1]=a2;
}
x0=pb[0][0];
y0=pb[1][0];
x2=xp(x0,y0,x0,y0,a,p);
y2=yp(x0,y0,x0,y0,a,p);
res1[0][0]=x0;
res1[1][0]=y0;
res1[0][1]=x2;
res1[1][1]=y2;
x1=x0;
y1=y0;
for(j=2;j<=k0-1;j++)
{
xw=x2-x1;
yw=y2-y1;
if(xw==0 && y1+y2==p )
{
res1[0][j]=-1000;
res1[1][j]=-1000;
```

```
printf("dot infinity");
res1[0][j+1]=res1[0][0];
res1[1][j+1]=res1[1][0];
res1[0][j+2]=res1[0][1];
res1[1][j+2]=res1[1][1];
y2=y0;
x2=x0;
x3= res1[0][1];
y3= res1[1][1];
j=j+2;
}
else if(!xw==0)
{
x3=xp(x1,y1,x2,y2,a,p);
y3=yp(x1,y1,x2,y2,a,p);
res1[0][j]=x3;
res1[1][j]=y3;
}
x1=x0;
x2=x3;

y1=y0;
y2=y3;
}

printf("k0*pb xpb(%d * %d)=%d ypb(%d *
%d)=%d\n",k0,pb[0][0],res1[0][k0-1],k0,pb[1][0],res1[1][k0-1]);
a1=res1[0][k0-1];
a2=res1[1][k0-1];
a3=562;
a4=201;
printf("x2s=%d y2s=%d\n",xp(a3,a4,a1,a2,a,p ), yp(a3,a4,a1,a2,a,p ) );
for(j=0;j<=nn-1;j++)
{
x1=pm[0][j];
y1=pm[1][j];
x2=res1[0][k0-1];
```

```
y2=res1[1][k0-1];
xw=x2-x1;
if(xw==0 && y1+y2==p)
{
shifr[j][2]=-1000;
shifr[j][3]=-1000;
printf("dot=infinity\n");

}
else
{
shifr[j][2]=xp(x1,y1,x2,y2,a,p);
shifr[j][3]=yp(x1,y1,x2,y2,a,p);
}
shifr[j][0]=shifr[0][0];
shifr[j][1]=shifr[0][1];
}

 for(i=0;i<=nol-1;i++)
{
        if(res[0][i]==pb[0][0] && res[1][i]==pb[1][0])
{
nb=i+1;
        printf(" **** number open kluch i=%d ******\n",nb);
}
}

x0=res[0][k0-1];
y0=res[1][k0-1];
x1=x0;
y1=y0;
de[0][0]=x0;
de[0][1]=y0;
x2=xp(x0,y0,x0,y0,a,p);
y2=yp(x0,y0,x0,y0,a,p);
de[1][0]=x2;
de[1][1]=y2;
```

```
for(j=2;j<=nb-1;j++)
{
xw=x2-x1;
if(xw==0 && y2+y1==p)
{
de[j][0]=-1000;
de[j][1]=-1000;
printf("dot infinity");
de[j+1][0]=x0;
de[j+1][1]=y0;
de[j+2][0]=de[1][0];
de[j+2][1]=de[1][1];
j=j+2;
}
else
{
x3=xp(x1,y1,x2,y2,a,p);
y3=yp(x1,y1,x2,y2,a,p);
de[0][j]=x3;
de[1][j]=y3;
}
x1=x0;

x2=x3;
y1=y0;
y2=y3;
}
for(j=0;j<=nn-1;j++)
{
x1=de[0][nb-1];
y1=-de[1][nb-1];
x2=shifr[j][2];
y2=shifr[j][3];
xw=x2-x1;
if(xw==0 && y1+y2==p)
{
otv[0][j]=-1000;
```

```
otv[1][j]=-1000;
printf("dot infinity");
}
else
{
otv[0][j]=xp(x1,y1,x2,y2,a,p);
otv[1][j]=yp(x1,y1,x2,y2,a,p);
}
}
printf("nb=%d\n",nb);
// shifrovanie//////////////////
for(j=0;j<=nn-1;j++)
{
printf(" **** soobchenie xm=%d ym=%d *******\n",pm[0][j],pm[1][j]);
printf("        ******* begin shifrovanie ********   \n");
printf("       ***** infinity haz number = %d ********\n",nol+1);
///end shifrovanie//////
///deshifrovanie/////////
xx51=otv[0][j];
xx52=otv[1][j];
printf("***** deshifrovanie *****\n");
printf("end x=%d end y=%d\n",xx51,xx52);
printf("proverka(%d %d)=%d\n",xx51,xx52,prov(xx51,xx52,a,b,p));
des[j]=xx51-xx52;
if(des[j]<0)
{
des[j]=des[j]+p;
}
printf("j=%d x=%d y=%d des=%d\n",j,xx51,xx52,des[j]);
/// end shifrovanie//////
}
printf("*****deshifrovanie****\n");

for(j=0;j<=nn-1;j++)
{
jj=des[j];
printf("%c \n",str[jj]);
```

```
}
for(j=0;j<=nn-1;j++)
{
jj=des[j];
        printf("j=%d,jj=%d\n",j,jj);
}
printf("shifr of text is:\n");
for(i=0;i<=nn-1;i++)
{

printf("[(%d,%d),(%d,%d)]\n",shifr[i][0],shifr[i][1],shifr[i][2],shifr[i][3]);

}
//////////////////// file.txt

remove("balka1.txt");
remove("balka2.txt");
file*file;

ofstream out("balka2.txt", ios::app);

file=fopen("balka2.txt","r");
out<<k0<<endl;
//fprintf(file,"%d",k0);
fclose(file);
printf("\n");
printf("//////////////////////");
int kk0=p;
printf("num(%d)=%d\n",kk0,nu(kk0));

printf("nachalo\n");////////////////////////////////////////////////////////////////////////////
/////////////
//return 0;
for(i=0;i<=nn-1;i++)
{
zyx(shifr[i][0],nol);
```

```
zyx(shifr[i][1],nol);
zyx(shifr[i][2],nol);
zyx(shifr[i][3],nol);
}
fclose(file);
 int ind,sad[4*nn-1],mm;
char arr00[nnn];
char rr[4*nn-1];
int  rf[4*nn+1][4],rs[4*nn+1];
ind=nu(nol);
file=fopen("balka1.txt","r");
if(file==null)
{
        printf(" not open file");
}
else
{
         for(i=0; i<=4*nn-1;i++)
{
fgets(arr00,nnn,file);

sad[i]=atoi(arr00);

         }

}
fclose(file);

for(i=0;i<=4*nn-1;i++)
{
        printf("i=%d l=%d\n", i ,sad[i]);
}
for(i=0;i<=4*nn-1;i++)
{
t=i%4;
t1=(i-i%4)/4;
```

```
shifr0[t1][t]=sad[i];
}
for(i=0;i<=nn-1;i++)
{
printf("i=%d shifr0(%d %d),(%d
%d)\n",i,shifr0[i][0],shifr0[i][1],shifr0[i][2],shifr0[i][3]);
}
x0=res[0][k0-1];
y0=res[1][k0-1];
x1=x0;
y1=y0;
de[0][0]=x0;
de[0][1]=y0;
x2=xp(x0,y0,x0,y0,a,p);
y2=yp(x0,y0,x0,y0,a,p);
de[1][0]=x2;
de[1][1]=y2;
 for(i=0;i<=nol-1;i++)
{
        if(res[0][i]==pb[0][0] && res[1][i]==pb[1][0])
{
nb=i+1;
        printf(" **** number open kluch i=%d ******\n",nb);
}
}

//////////////////////////////////////

for(j=2;j<=nb-1;j++)
{
xw=x2-x1;
if(xw==0 && y2+y1==p)
{
de[j][0]=-1000;
de[j][1]=-1000;
printf("dot infinity");
de[j+1][0]=x0;
```

```
de[j+1][1]=y0;
de[j+2][0]=de[1][0];
de[j+2][1]=de[1][1];
j=j+2;
}
else
{
x3=xp(x1,y1,x2,y2,a,p);
y3=yp(x1,y1,x2,y2,a,p);
de[0][j]=x3;
de[1][j]=y3;
}
x1=x0;

x2=x3;
y1=y0;
y2=y3;
}
for(j=0;j<=nn-1;j++)
{
x1=de[0][nb-1];
y1=-de[1][nb-1];
x2=shifr0[j][2];
y2=shifr0[j][3];
xw=x2-x1;
if(xw==0 && y1+y2==p)
{
otv[0][j]=-1000;
otv[1][j]=-1000;
printf("dot infinity");
}
else
{
otv[0][j]=xp(x1,y1,x2,y2,a,p);
otv[1][j]=yp(x1,y1,x2,y2,a,p);
}
}
```

```
printf("nb=%d\n",nb);
// shifrovanie//////////////////
for(j=0;j<=nn-1;j++)
{
printf(" **** soobchenie0 xm0=%d ym0=%d
********\n",pm[0][j],pm[1][j]);
printf("         ******** begin shifrovanie0 *********   \n");
printf("       ****** infinity haz number = %d *********\n",nol+1);
///end shifrovanie//////

///deshifrovanie/////////
xx51=otv[0][j];
xx52=otv[1][j];
printf("****** deshifrovanie0 ******\n");
printf("end x0=%d end y0=%d\n",xx51,xx52);
printf("proverka0(%d %d)=%d\n",xx51,xx52,prov(xx51,xx52,a,b,p));
des[j]=xx51-xx52;
if(des[j]<0)
{
des[j]=des[j]+p;
}
printf("j=%d x0=%d y0=%d des0=%d\n",j,xx51,xx52,des[j]);
/// end shifrovanie//////
}
printf("******deshifrovanie0****\n");

for(j=0;j<=nn-1;j++)
{
jj=des[j];
printf("%c \n",str[jj]);

}
for(j=0;j<=nn-1;j++)
{
jj=des[j];
        printf("j=%d,jj=%d\n",j,jj);
}
```

```
printf("shifr0 of text is:\n");
for(i=0;i<=nn-1;i++)
{
printf("[s=%d
kg:(%d,%d),pm+k*pb:(%d,%d)]\n",i,shifr0[i][0],shifr0[i][1],shifr0[i][2],s
hifr0[i][3]);
}
printf("end\n");

return 0;
                                                    }
```

Приложение 3
Программа хеш - функции

```
        #include<stdio.h>
#include<math.h>
 int prog3( int t);
int main()
{
        int m,n,m0,n0,res;
        n=123;
        n0=prog3(n);
        printf("n0=%d\n",n0);
        int bac[n0],k,i;
        k=n;
        m=0;
        for(i=1;i<=n0;i++)
        {

                bac[i]=k%10;
                 printf("bac(%d)=%d\n",i,bac[i]);
                k=(k-k%10)/10;
                m=m+bac[i]*pow(10,n0-i);
        }
         printf("m=%d\n",m);
         if(n<=m)
         {
                res=m%n;
         }
         else
         {
                res=n%m;
         }
         printf("hesch(%d)=%d\n",n,res);
}
int prog3( int t)
{
 int i,t1,sss;
t1=t;
sss=0;
for(i=0;i<=100;i++)
```

```
{
if(t1>=10)
{
t1=(t1-t1%10)/10;
sss=sss+1;
}
else if(t1<10)
{
t1=t1;
sss=sss+1;
i=100;
}
}
return sss;
}
```

Приложение 4

Программа библиотеки интерфейса dll, выполненная на языке Compaq Visual Fortran 6.6

```
MODULE Primer

USE Xeffort
USE Xflogm
USE DFWIN
USE KERNEL32
IMPLICIT NONE

INCLUDE 'Resource.fd'
INTEGER(4):: ICOLOR,ICOLOR1,IBKCOLOR,IBKCOLOR1,IBKCOLOR2
INTEGER(4):: a,b,p,z1,z2,k
INTEGER(HANDLE):: HINSTANCE,HTHREADWORKER
!=================================================
===============
CONTAINS
!=================================================
===============
!PURPOSE: Initialization function called by XFT library on app start.
LOGICAL FUNCTION XInit(szCmdLine,nCmdShow)
!DEC$IF (_DF_VERSION_.GE.650)
!DEC$ATTRIBUTES DEFAULT, DECORATE, ALIAS: 'XINIT':: XInit
!DEC$ELSE
!DEC$ATTRIBUTES ALIAS: '_XINIT@12':: XInit
!DEC$ENDIF

IMPLICIT NONE

CHARACTER(*), INTENT(IN):: szCmdLine
INTEGER, INTENT(IN):: nCmdShow

LOGICAL:: bSt,FLAG


IBKCOLOR=#CC3299
ICOLOR=#00FFFF
```

IBKCOLOR1=(#D7EBFA) !(#87B8DE)
ICOLOR1=#FF0000

IBKCOLOR2=(#EEEEAF) !(#CBC0FF) !(#D8BFD8) !(#DDA0DD) !
!(#D7EBFA)

IF (XCreateDialogApp(IDD_MAIN, idIcon=IDI_ICON_MAIN)) THEN
!Todo: Initialize dialog controls here via XCtlSet(Sub)
FLAG=DLGSET(XW_FRAME,IDD_MAIN,IBKCOLOR,DLG_BKCOLOR)

!End Todo: Initialize dialog controls here via XCtlSet(Sub)
FLAG=DLGSETSUB(XW_FRAME,PUSK,PUSKSUB_THREAD)
FLAG=DLGSETSUB(XW_FRAME,PUSK2,STOP_SUB)
FLAG=DLGSETSUB(XW_FRAME,PUSK3,CLEAR_SUB)

FLAG=DLGSETSUB(XW_FRAME,IDC_EDIT2,SUB_EDIT2,DLG_CHANG
E)
FLAG=DLGSETSUB(XW_FRAME,IDD_MAIN,SUB_MAIN)

FLAG=DLGSET(XW_FRAME,IDD_MAIN,IBKCOLOR,DLG_BKCOLOR)

FLAG=DLGSET(XW_FRAME,IDC_STATIC1,IBKCOLOR,DLG_BKCOLOR
)

FLAG=DLGSET(XW_FRAME,IDC_STATIC2,IBKCOLOR,DLG_BKCOLOR
)

FLAG=DLGSET(XW_FRAME,IDC_STATIC3,IBKCOLOR,DLG_BKCOLOR
)

FLAG=DLGSET(XW_FRAME,IDC_STATIC4,IBKCOLOR,DLG_BKCOLOR
)

FLAG=DLGSET(XW_FRAME,IDC_STATIC5,IBKCOLOR,DLG_BKCOLOR
)

FLAG=DLGSET(XW_FRAME,IDC_STATIC6,IBKCOLOR,DLG_BKCOLOR
)

FLAG=DLGSET(XW_FRAME,IDC_STATIC7,IBKCOLOR,DLG_BKCOLOR
)

FLAG=DLGSET(XW_FRAME,IDC_STATIC8,IBKCOLOR,DLG_BKCOLOR
)

FLAG=DLGSET(XW_FRAME,IDC_STATIC9,IBKCOLOR,DLG_BKCOLOR
)

FLAG=DLGSET(XW_FRAME,IDC_STATIC10,IBKCOLOR,DLG_BKCOLO
R)

FLAG=DLGSET(XW_FRAME,IDC_STATIC11,IBKCOLOR,DLG_BKCOLO
R)

FLAG=DLGSET(XW_FRAME,IDC_STATIC12,IBKCOLOR,DLG_BKCOLO
R)

FLAG=DLGSET(XW_FRAME,IDC_STATIC13,IBKCOLOR,DLG_BKCOLO
R)

FLAG=DLGSET(XW_FRAME,IDC_STATIC14,IBKCOLOR,DLG_BKCOLO
R)

FLAG=DLGSET(XW_FRAME,IDC_STATIC1,ICOLOR,DLG_COLOR)
FLAG=DLGSET(XW_FRAME,IDC_STATIC2,ICOLOR,DLG_COLOR)
FLAG=DLGSET(XW_FRAME,IDC_STATIC3,ICOLOR,DLG_COLOR)
FLAG=DLGSET(XW_FRAME,IDC_STATIC4,ICOLOR,DLG_COLOR)
FLAG=DLGSET(XW_FRAME,IDC_STATIC5,ICOLOR,DLG_COLOR)
FLAG=DLGSET(XW_FRAME,IDC_STATIC6,ICOLOR,DLG_COLOR)
FLAG=DLGSET(XW_FRAME,IDC_STATIC7,ICOLOR,DLG_COLOR)
FLAG=DLGSET(XW_FRAME,IDC_STATIC8,ICOLOR,DLG_COLOR)
FLAG=DLGSET(XW_FRAME,IDC_STATIC9,ICOLOR,DLG_COLOR)
FLAG=DLGSET(XW_FRAME,IDC_STATIC10,ICOLOR,DLG_COLOR)
FLAG=DLGSET(XW_FRAME,IDC_STATIC11,ICOLOR,DLG_COLOR)
FLAG=DLGSET(XW_FRAME,IDC_STATIC12,ICOLOR,DLG_COLOR)
FLAG=DLGSET(XW_FRAME,IDC_STATIC13,ICOLOR,DLG_COLOR)

FLAG=DLGSET(XW_FRAME,IDC_STATIC14,ICOLOR,DLG_COLOR)

FLAG=DLGSET(XW_FRAME,IDC_EDIT1,ICOLOR1,DLG_COLOR)
FLAG=DLGSET(XW_FRAME,IDC_EDIT2,ICOLOR1,DLG_COLOR)
FLAG=DLGSET(XW_FRAME,IDC_EDIT3,ICOLOR1,DLG_COLOR)
FLAG=DLGSET(XW_FRAME,IDC_EDIT4,ICOLOR1,DLG_COLOR)
FLAG=DLGSET(XW_FRAME,IDC_EDIT5,ICOLOR1,DLG_COLOR)
FLAG=DLGSET(XW_FRAME,IDC_EDIT6,ICOLOR1,DLG_COLOR)
FLAG=DLGSET(XW_FRAME,IDC_EDIT7,ICOLOR1,DLG_COLOR)
FLAG=DLGSET(XW_FRAME,IDC_EDIT8,ICOLOR1,DLG_COLOR)
FLAG=DLGSET(XW_FRAME,IDC_EDIT9,ICOLOR1,DLG_COLOR)
FLAG=DLGSET(XW_FRAME,IDC_EDIT10,ICOLOR1,DLG_COLOR)
FLAG=DLGSET(XW_FRAME,IDC_EDIT11,ICOLOR1,DLG_COLOR)

FLAG=DLGSET(XW_FRAME,IDC_EDIT1,IBKCOLOR2,DLG_BKCOLOR)

FLAG=DLGSET(XW_FRAME,IDC_EDIT2,IBKCOLOR2,DLG_BKCOLOR)

FLAG=DLGSET(XW_FRAME,IDC_EDIT3,IBKCOLOR2,DLG_BKCOLOR)

FLAG=DLGSET(XW_FRAME,IDC_EDIT4,IBKCOLOR1,DLG_BKCOLOR)

FLAG=DLGSET(XW_FRAME,IDC_EDIT5,IBKCOLOR1,DLG_BKCOLOR)

FLAG=DLGSET(XW_FRAME,IDC_EDIT6,IBKCOLOR1,DLG_BKCOLOR)

FLAG=DLGSET(XW_FRAME,IDC_EDIT7,IBKCOLOR1,DLG_BKCOLOR)

FLAG=DLGSET(XW_FRAME,IDC_EDIT8,IBKCOLOR1,DLG_BKCOLOR)

FLAG=DLGSET(XW_FRAME,IDC_EDIT9,IBKCOLOR1,DLG_BKCOLOR)

FLAG=DLGSET(XW_FRAME,IDC_EDIT10,IBKCOLOR1,DLG_BKCOLOR
)

```
FLAG=DLGSET(XW_FRAME,IDC_EDIT11,IBKCOLOR2,DLG_BKCOLOR
)

!-----------------------------------------------
FLAG=DLGSET(XW_FRAME,IDC_EDIT2,'polotsk 2016')
FLAG=DLGSET(XW_FRAME,IDC_EDIT1,'96587')

FLAG=DLGSET(XW_FRAME,IDC_EDIT4,'-1')
FLAG=DLGSET(XW_FRAME,IDC_EDIT5,'188')
FLAG=DLGSET(XW_FRAME,IDC_EDIT6,'751')
FLAG=DLGSET(XW_FRAME,IDC_EDIT7,'201')
FLAG=DLGSET(XW_FRAME,IDC_EDIT8,'5')
FLAG=DLGSET(XW_FRAME,IDC_EDIT9,'17')
!FLAG=DLGSET(XW_FRAME,IDC_EDIT10,'12')
!-----------------------------------------------


IF (XModalDialog(XW_FRAME).EQ.IDOK) THEN
!Todo: place logic after dialog is exited via OK button here (XCtlGet)




END IF

END IF

!For a dialog-bassed application, the XInit may return .FALSE.
!Alternatively, use XModelessDialog, but return .TRUE. and set
!handlers for IDOK/IDCANCEL buttons.
XInit = .FALSE.

END FUNCTION XInit
!=================================================
===============
SUBROUTINE PUSKSUB_THREAD(DLG,C_NAME,CBTYPE)
IMPLICIT NONE
TYPE(DIALOG):: DLG
INTEGER(4),INTENT(IN):: C_NAME,CBTYPE
```

```
LOGICAL:: FLAG
TYPE (T_SECURITY_ATTRIBUTES),POINTER:: NULL_SA
INTEGER(4):: idTHREADWORKER
HTHREADWORKER = CreateThread( NULL_SA, 0,LOC(PUSKSUB),   &
                LOC(dlg), 0 , LOC(idTHREADWORKER) )

FLAG=SetThreadPriority(HTHREADWORKER,THREAD_PRIORITY_NOR
MAL)
ENDSUBROUTINE PUSKSUB_THREAD
!=================================================
===============
SUBROUTINE STOP_SUB(DLG,C_NAME,CBTYPE)
IMPLICIT NONE
TYPE(DIALOG):: DLG
INTEGER(4),INTENT(IN):: C_NAME,CBTYPE
LOGICAL:: FLAG
TYPE (T_SECURITY_ATTRIBUTES),POINTER:: NULL_SA
INTEGER(4):: idTHREADWORKER
CALL  CLOSE_LIBRARY(HINSTANCE);
FLAG=TERMINATETHREAD(HTHREADWORKER,0)
IF(FLAG) THEN
FLAG=DLGSET(DLG,IDC_STATIC12,'Process terminated succesfully!')
ELSE
FLAG=DLGSET(DLG,IDC_STATIC12,'Process not terminated!
Unsuccesfully. ')
ENDIF

ENDSUBROUTINE STOP_SUB
!=================================================
===============



SUBROUTINE CLEAR_SUB(DLG,C_NAME,CBTYPE)
IMPLICIT NONE
TYPE(DIALOG):: DLG
INTEGER(4),INTENT(IN):: C_NAME,CBTYPE
LOGICAL:: FLAG
TYPE (T_SECURITY_ATTRIBUTES),POINTER:: NULL_SA
INTEGER(4):: idTHREADWORKER

FLAG=DLGSET(DLG,IDC_EDIT1,' ')
```

```
FLAG=DLGSET(DLG,IDC_EDIT3,' ')
FLAG=DLGSET(DLG,IDC_STATIC12,' ')
FLAG=DLGSET(DLG,IDC_EDIT11,' ')

ENDSUBROUTINE CLEAR_SUB
!=====================================================
===============
SUBROUTINE SUB_EDIT2(DLG,C_NAME,CBTYPE)
IMPLICIT NONE
TYPE(DIALOG):: DLG
INTEGER(4),INTENT(IN):: C_NAME,CBTYPE
LOGICAL:: FLAG
CHARACTER(LEN=1000):: STROKA
INTEGER(4):: LENTH
FLAG=DLGSET(DLG,IDC_EDIT10,' ')
FLAG=DLGGET(DLG,IDC_EDIT2,STROKA)
LENTH=LEN_TRIM(STROKA)
WRITE(STROKA,*) LENTH
STROKA=ADJUSTL(STROKA)
FLAG=DLGSET(DLG,IDC_EDIT10,TRIM(STROKA))
ENDSUBROUTINE SUB_EDIT2
!=====================================================
===============
SUBROUTINE SUB_MAIN(DLG,C_NAME,CBTYPE)
IMPLICIT NONE
TYPE(DIALOG):: DLG
INTEGER(4),INTENT(IN):: C_NAME,CBTYPE
LOGICAL:: FLAG
CHARACTER(LEN=1000):: STROKA
INTEGER(4):: LENTH
CALL SUB_EDIT2(DLG,C_NAME,CBTYPE)
ENDSUBROUTINE SUB_MAIN


!=====================================================
===============

SUBROUTINE PUSKSUB(DLG,C_NAME,CBTYPE)
!USE ISO_C_BINDING
IMPLICIT NONE
TYPE(DIALOG):: DLG
INTEGER(4),INTENT(IN):: C_NAME,CBTYPE
```

```
LOGICAL:: FLAG
CHARACTER(LEN=80)::STRING
CHARACTER(LEN=10000)::STRING_OUT,STRING_CIFER
CHARACTER(LEN=1000)::STR
CHARACTER(LEN=1):: CHAR1
INTEGER(4):: J,I
INTEGER(4):: nDLL_C_PLUS_PLUS
INTEGER(4):: a,b,p,z1,z2,k
CHARACTER(LEN=1000)::STR1
CHARACTER(LEN=1000)::STR_DLL
INTEGER(4)::shifr(0:1000,0:3),shifr0(0:1000,0:3)
INTEGER(4):: n,nn,nnn
INTEGER(4)::n7,nn97,nnn7
INTEGER(4):: t

INTERFACE
INTEGER FUNCTION NU_DLL(N1)
INTEGER ,INTENT(IN):: N1
ENDFUNCTION
END INTERFACE


INTERFACE
INTEGER FUNCTION fort_nu(N1)
INTEGER ,INTENT(IN):: N1
ENDFUNCTION
END INTERFACE

INTERFACE
SUBROUTINE fnDLL_C_PLUS_PLUS
ENDSUBROUTINE
END INTERFACE

INTERFACE
SUBROUTINE fn_main
ENDSUBROUTINE
END INTERFACE


POINTER(FARPROC0,fnDLL_C_PLUS_PLUS)
POINTER(FARPROC1,a)
```

```
POINTER(FARPROC2,b)
POINTER(FARPROC3,p)
POINTER(FARPROC4,z1)
POINTER(FARPROC5,z2)
POINTER(FARPROC6,k)
POINTER(FARPROC7,str1)
POINTER(FARPROC8,nn)
POINTER(FARPROC9,shifr)
POINTER(FARPROC10,shifr0)

POINTER(FARPROC15,NU_DLL)
POINTER(FARPROC20,nDLL_C_PLUS_PLUS)
POINTER(FARPROC25,fn_main)


POINTER(FARPROC30,n7)
POINTER(FARPROC35,nn97)
POINTER(FARPROC40,nnn7)

POINTER(FARPROC45,fort_nu)
POINTER(FARPROC50,STR_DLL)



HINSTANCE=LoadLibrary("DLL_C_PLUS_PLUS"C)
                    IF(HINSTANCE/=0) THEN
FLAG=DLGSET(XW_FRAME,IDC_STATIC12,'Длл успешно загружена!')
                    CONTINUE
                    ELSE
FLAG=DLGSET(XW_FRAME,IDC_STATIC12,'Длл не загружена!')
CALL  CLOSE_LIBRARY(HINSTANCE); RETURN ;
                    ENDIF


FARPROC0=GetProcAddress(HINSTANCE,"?fnDLL_C_PLUS_PLUS@@Y
AXXZ"C)
FARPROC1=GetProcAddress(HINSTANCE,"?a@@3HA"C) !a
FARPROC2=GetProcAddress(HINSTANCE,"?b@@3HA"C) !b
FARPROC3=GetProcAddress(HINSTANCE,"?p@@3HA"C) !p
FARPROC4=GetProcAddress(HINSTANCE,"?z1@@3HA"C) !z1
FARPROC5=GetProcAddress(HINSTANCE,"?z2@@3HA"C) !z2
FARPROC6=GetProcAddress(HINSTANCE,"?k@@3HA"C)  !k
```

```
FARPROC7=GetProcAddress(HINSTANCE,"?str1@@3PADA"C) !str1
FARPROC8=GetProcAddress(HINSTANCE,"?nn@@3HA"C) !nn
FARPROC9=GetProcAddress(HINSTANCE,"?shifr@@3PAY03HA"C)
!shifr
FARPROC10=GetProcAddress(HINSTANCE,"?shifr0@@3PAY03HA"C)
!shifr0

FARPROC15=GetProcAddress(HINSTANCE,"?nu@@YAHH@Z"C) !shifr0

FARPROC20=GetProcAddress(HINSTANCE,"?nDLL_C_PLUS_PLUS@@3H
A"C) !shifr0
FARPROC25=GetProcAddress(HINSTANCE,"?fn_main@@YAXXZ"C)
!shifr0


FARPROC30=GetProcAddress(HINSTANCE,"?n7@@3HA"C) !shifr0
FARPROC35=GetProcAddress(HINSTANCE,"?nn97@@3HA"C) !shifr0
FARPROC40=GetProcAddress(HINSTANCE,"?nnn7@@3HA"C) !shifr0

FARPROC45=GetProcAddress(HINSTANCE,"?fort_nu@@YAHXZ"C)
!shifr0
FARPROC50=GetProcAddress(HINSTANCE,"?str10@@3PADA"C) !shifr0

FLAG=DLGSET(DLG,IDC_STATIC12,' ')



FLAG=DLGGET(DLG,IDC_EDIT2,str1)

FLAG=DLGGET(DLG,IDC_EDIT4,STRING)
READ(STRING,'(I10)',ERR=1) a
FLAG=DLGGET(DLG,IDC_EDIT5,STRING)
READ(STRING,'(I10)',ERR=1) b
FLAG=DLGGET(DLG,IDC_EDIT6,STRING)
READ(STRING,'(I10)',ERR=1) p

FLAG=DLGGET(DLG,IDC_EDIT7,STRING)
READ(STRING,'(I10)',ERR=1) z1
FLAG=DLGGET(DLG,IDC_EDIT8,STRING)
READ(STRING,'(I10)',ERR=1)z2
FLAG=DLGGET(DLG,IDC_EDIT9,STRING)
READ(STRING,'(I10)',ERR=1) k
```

```
FLAG=DLGGET(DLG,IDC_EDIT10,STRING)
READ(STRING,'(I10)',ERR=1) nn


IF(NN<1) THEN
FLAG=DLGSET(DLG,IDC_STATIC12,'Error parameter nn.Program
terminated. ')
RETURN
ENDIF

FLAG=(FARPROC0/=0).AND.(FARPROC1/=0).AND.(FARPROC2/=0).AND.
(FARPROC3/=0).AND.(FARPROC4/=0).AND.(FARPROC5/=0).AND. &
(FARPROC6/=0).AND.(FARPROC7/=0).AND.(FARPROC8/=0).AND.(FARP
ROC9/=0).AND.(FARPROC10/=0).AND.(FARPROC15/=0) &
.AND.(FARPROC20/=0).AND.(FARPROC25/=0)

                IF(FLAG) THEN
                CONTINUE
                ELSE
        FLAG=DLGSET(XW_FRAME,IDC_STATIC12,'Не все адреса
                процедур получены!')
        CALL  CLOSE_LIBRARY(HINSTANCE); RETURN ;
                ENDIF



!FLAG=DLGGET(DLG,IDC_EDIT1,STRING)
!READ(STRING,'(I10)',ERR=1) nDLL_C_PLUS_PLUS
!I=fort_nu(nDLL_C_PLUS_PLUS)
!WRITE(STRING,*) I; STRING=ADJUSTL(STRING);
!FLAG=DLGSET(XW_FRAME,IDC_STATIC12,'NU='//TRIM(STRING))
!CALL fnDLL_C_PLUS_PLUS()
nnn7=80 ;
n7=800 ;
!nn9=1000;
CALL fn_main()


STRING_OUT=' ';STRING_CIFER=' ';
DO J=1,4*NN,1
```

```
WRITE(STR,*) shifr(J-1,0)
STR=ADJUSTL(STR)

STRING_CIFER=TRIM(STRING_CIFER)//'  '//TRIM(STR)
WRITE(STR,*) shifr0(J-1,0)
STR=ADJUSTL(STR)
STRING_OUT=TRIM(STRING_OUT)//'  '//TRIM(STR)
ENDDO
STRING_CIFER=ADJUSTL(STRING_CIFER)
STRING_OUT=ADJUSTL(STRING_OUT)
FLAG=DLGSET(XW_FRAME,IDC_EDIT1,TRIM(STRING_CIFER))
FLAG=DLGSET(XW_FRAME,IDC_EDIT3,TRIM(STRING_OUT))
STR_DLL=ADJUSTL(STR_DLL)
FLAG=DLGSET(XW_FRAME,IDC_EDIT11,TRIM(STR_DLL))

FLAG=DLGSET(XW_FRAME,IDC_STATIC12,'Finished!')


RETURN
1 FLAG=DLGSET(DLG,IDC_STATIC12,'Error input data')
CALL  CLOSE_LIBRARY(HINSTANCE);

ENDSUBROUTINE PUSKSUB
!=================================================
===============
SUBROUTINE CLOSE_LIBRARY(HINSTANCE)
USE KERNEL32
IMPLICIT NONE
INTEGER(HANDLE),INTENT(IN):: HINSTANCE
LOGICAL:: FLAG
FLAG=FreeLibrary(HINSTANCE)
IF(FLAG) THEN
FLAG=CLOSEHANDLE(HINSTANCE)
FLAG=DLGSET(XW_FRAME,IDC_STATIC12,'Длл отключена!')
!CALL SLEEPQQ(100)
ENDIF
END SUBROUTINE CLOSE_LIBRARY
!=================================================
===============

END MODULE Primer
```

Пастухов Дмитрий Феликсович
Пастухов Юрий Феликсович
Павел Рональдович Синица
Полоцкий государственный университет
Новополоцк 2016