Правительство Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»
(НИУ ВШЭ)

Московский институт электроники и математики им. А.Н.Тихонова

ОТЧЕТ
О ПРАКТИЧЕСКОЙ РАБОТЕ № 6

по дисциплине «Криптографические методы защиты информации»

ТЕМА РАБОТЫ

СХЕМЫ ЭЛЕКТРОННОЙ ПОДПИСИ. ГОСТ 34.10-2012

Студент гр. МКБ231

_____

«___» _____ 2024 г.

Руководитель
Заведующий кафедрой информационной безопасности киберфизических систем
канд. техн. наук, доцент

_____

«___» _____ 2024 г.

Москва 2024

СОДЕРЖАНИЕ

1. Задание на практическую работу

Целью работы является создание программной реализации схемы электронной цифровой подписи, представленной в ГОСТ 34.10-2012.

В рамках практической работы необходимо выполнить следующее:

1. Написать программную реализацию схемы электронной цифровой подписи ГОСТ 34.10-2012

   - принимать на вход файл, содержащий открытый текст для подписания или проверки электронной подписи

   - принимать на вход файл, содержащий электронную цифровую подпись

   - принимать на вход ключ подписи или ключ проверки подписи

   - предоставлять возможность генерации ключевой пары и параметров

   - осуществлять проверку или формирование электронной цифровой подписи по выбору пользователя.

2. Составить описание работы алгоритма формирования и проверки электронной цифровой подписи по ГОСТ 34.10-2012
3. Составить описание программной реализации с учетом особенностей выбранной среды разработки и языка программирования.
4. Включить в отчет результаты работы программы
5. Предоставить выводы о проделанной работе

2. Описание схемы электронной подписи ГОСТ 34.10-2012

ГОСТ 34.10-2012 является действующим стандартом для электронной подписи в РФ. ГОСТ 34.10-2012 оперирует ЭЦП размером 512 или 1024 бит. Для хэширования сообщения используется функция, определенная в ГОСТ 34.11-2012.
Схема электронной цифровой подписи, представленная в ГОСТ Р 34.10-2012, оперирует следующими параметрами:
Простое число p – модуль эллиптической кривой.
Эллиптическая кривая E, задаваемая своим инвариантом J(E) или коэффициентами a, b.
Целое число m – порядок группы точек эллиптической кривой E, такое, что $(p+1-2\sqrt{p}) \le m \le (p+1+2\sqrt{p})$.
Простое число q – порядок циклической подгруппы группы точек эллиптической кривой E, для которого выполнены следующие условия: m= n*q, n ∈ N, где
2^254 < q < 2^256 или 2^508 < q < 2^512 в зависимости от схемы хэширования (256 или 512 бит)
Точка P ≠ O эллиптической кривой E с координатами (Xp; Yp), удовлетворяющая равенству q*P=O.
Хэш-функция $h(\cdot) : V^* \rightarrow V_l$, где l=256, 512.
Ключ подписи – целое число d.
Ключ проверки подписи – точка эллиптической кривой Q =d*P.

2.1 Описание процедуры формирования ЭЦП

1. Вычислить хэш-код сообщения M: $\overline{h} = h(M)$.
2. Вычислить целое число a, двоичным представлением которого является вектор $\overline{h}$, и определить $e = a \pmod q$. Если $e = 0$, то определить $e = 1$.
3. Сгенерировать случайное целое число k, удовлетворяющее неравенству $0 < k < q$.
4. Вычислить точку эллиптической кривой $C = kP$ и определить $r = x_C \pmod q$. Если $r = 0$, то вернуться к шагу 3
5. Вычислить значение $s = (rd + ke) \pmod q$. Если $s = 0$, то вернуться к шагу 3.
6. Вычислить двоичные векторы, соответствующие числам r и s, и определить цифровую подпись $\zeta = \overline{r} \parallel \overline{s}$ как конкатенацию данных двоичных векторов.

2.2Описание процедуры проверки ЭЦП

1. По полученной подписи ζ вычислить целые числа r и s. Если выполнены неравенства $0 < r < q$, $0 < s < q$, то перейти к следующему шагу. В противном случае подпись неверна.
2. Вычислить хэш-код сообщения M: $\overline{h} = h(M)$.
3. Вычислить целое число a, двоичным представлением которого является вектор $\overline{h}$, и определить $e = a \pmod q$. Если $e = 0$, то определить $e = 1$.
4. Вычислить значение $v = e^{-1} \pmod q$.
5. Вычислить значения $z_1 = sv \pmod q$, $z_2 = -rv \pmod q$.
6. Вычислить точку эллиптической кривой $C = z_1P + z_2Q$ и определить значение $R = x_C \pmod q$.
7. Если выполнено равенство $R = r$, то подпись принимается, в противном случае, подпись неверна.

2.3 Описание процедуры вычисления хэш-кода сообщения

Для вычисления хэш-кода сообщения используется алгоритм «Стрибог», определенный в ГОСТ 34.11-2012

Исходными данными для процедуры вычисления хэш-кода H(M) является подлежащее хэшированию сообщение $M \in \mathrm{V}^*$ и $IV \in \mathrm{V}_{512}$-инициализаторный вектор .

Алгоритм вычисления функции H состоит из следующих этапов.
Этап 1
Присвоить начальные значения текущих величин:
1.1. h := IV,
1.2. $N := 0^{512} \in V_{512}$;
1.3. $\Sigma := 0^{512} \in V_{512}$;
1.4. Перейти к этапу 2.
Этап 2
2.1. Проверить условие |M| < 512.
При положительном исходе перейти к этапу 3.
В противном случае выполнить последовательность вычислений по 2.2 - 2.7.
2.2. Вычислить подвектор $m \in \mathrm{V}_{512}$ сообщения M: M = M'||m. Далее выполнить последовательность вычислений:
2.3. $h := g_{\mathrm{N}}(h, m)$.
2.4. $N := \mathrm{Vec}_{512}\left(\mathrm{Int}_{512}(N)(\ 512\right)$.
2.5. $\Sigma := \mathrm{Vec}_{512}\left(\mathrm{Int}_{512}(\Sigma)(\ \mathrm{Int}_{512}(m)\right)$.
2.6. M = M'
2.7. Перейти к шагу 2.1.
Этап 3
3.1. $m := 0^{511-|\mathrm{M}|} \| 1 \| M$.
3.2. $h : g_{\mathrm{N}}(h, m)$.
3.3. $N := \mathrm{Vec}_{512}\left(\mathrm{Int}_{512}(N)(\ |M|\right)$.
3.4. $\Sigma := \mathrm{Vec}_{512}\left(\mathrm{Int}_{512}(\Sigma)(\ \mathrm{Int}_{512}(m)\right)$.
3.5. $h : g_0(h, N)$.
3.6.

$$h := \begin{cases} g_0(h, \Sigma), & \text{для функции хэширования с длиной хэш-кода 512 бит;} \\ MSB_{256}\left(g_0(h, \Sigma)\right), & \text{для функции хэширования с длиной хэш-кода 256 бит.} \end{cases}$$

3. Реализация алгоритма ГОСТ 34.10-2012

3.1 Общие сведения

Программа осуществления подписания файла электронной цифровой подписью и проверки этой подписи согласно ГОСТ 34.10-2012 реализована на языке программирования C# в виде консольного приложения с использованием IDE Visual Studio 2022 и включает в себя пять файлов: Program.cs, PreSet.cs, GetHash.cs, Init.cs, Point.cs.

3.2 Описание работы программы

1. Инициализация параметров:
   - генерируется простое число q длиной 254 или 508 бит, порядок циклической группы
   - используя q и значение множителя n > 65537 получаем значение m = p*q, на основании которого осуществляется генерация простого числа p – модуля эллиптической кривой
   - выбираем значение инварианта IV=1696 вычисляем значения коэффициентов эллиптической кривой a и b
   - генерируется ключ подписи d
   - генерируются координаты точки P
   - полученные значения заносятся в файлы для последующего использования
2. Формирование электронной цифровой подписи
   - вычисляется хеш-функция исходного сообщения
   - на основе инициализированных параметров и значения хэш исходного сообщения формируется электронная подпись в виде вектора
   - вычисляются координаты точки Q – ключа проверки электронной подписи
   - сообщение, электронная подпись, ключ проверки и необходимые параметры передаются получателю
3. Проверка электронной подписи
   - проверяется первичная корректность электронной подписи
   - вычисляется хэш-функция сообщения
   - проверяется истинность электронной подписи и истинность исходного сообщения с использованием переданных параметров, сообщения и ключа проверки электронной подписи.
4. Программа реализована в консольном режиме, возможны три варианта использования:
   - инициализация параметров и генерация ключа подписи.
   - генерация электронной подписи для исходного сообщения
   - проверка истинности электронной подписи и исходного сообщения
5. Список классов и методов, реализованных в программе:

```
- class PreSet – содержит постоянные вектора и матрицы, определенные ГОСТ 34.11-2012 для использования при вычислении хэш-функции

– class GetHash – содержит хэш-значение и методы для его вычисления

– GetHash(byte[] Message, int hash_size) – метод, который используется в качестве конструктора (устанавливает значение хэш)
```

- byte[] Add_2(byte[] a, byte[] b) – функция сложения векторов a и b по модулю 2

- byte[] Add_512(byte[] a, byte[] b) – функция сложения векторов a и b в кольце вычетов по модулю 2 в степени n

- byte[] TurnS(byte[] a) – функция осуществляет нелинейное биективное преобразование S 512 битного блока a

- byte[] TurnP(byte[] a) – функция осуществляет перестановку байт 512 битного блока a (преобразование P)

- byte[] TurnL(byte[] a) – функция осуществляет линейное преобразование L 512-бит блока a

- byte[] GetK(byte[] a, int i) – функция осуществляет преобразование E используя сообщение m и сессионный ключ k

- byte[] TurnG(byte[] h, byte[] N, byte[] m) – функция сжатия G, возвращает значение hash

- byte[] FillTail(byte[] m) – функция возвращает "хвост" сообщения, приведенного до длины в 512 бит

- byte[] FillBytes(int len, byte val) – функция заполняет массив байт длиной len значениями val

- ulong UlongFromBytes(byte[] input, int index) – функция возвращает 64 битовое число из массива байт[8]

- byte[] BytesFromUlong(ulong input) – функция раскладывает 64 битное число в массив байт[8]

- class Init – класс, содержит переменные p, q, a, b, x, y, d и методы по их созданию(модификации)

- Init(int length) – метод-конструктор инициализации параметров для заданной длине хэш значения

- BigInteger GenD(BigInteger q, BigInteger p) – функция генерирует число-вычет по модулю p не больше q

- BigInteger NextPrime(int length, int mr) – функция возвращает простое число длиной length бит

- BigInteger PrevPrime(BigInteger n, int mr) – функция возвращает первое простое число меньшее n

- GenAB(int iv) – метод вычисляет значение коэффициентов a и b эллиптической кривой по известному инварианту и модулю

- BigInteger GenBigInteger(int length) – функция создает случайное число заданной длиной бит

- bool IsPrime(BigInteger n, int mr) – функция тест Миллера-Рабина на простоту числа с числом раундов k

- BigInteger Legendre(BigInteger a, BigInteger p) – функция вычисляет символ Лежандра

- BigInteger Sqrt(BigInteger bi) – функция возвращает квадратный корень из большого числа

- BigInteger SqrtMod(BigInteger a, BigInteger p) – функция возвращает квадратный корень числа a по модулю p

- BigInteger GetReverse(BigInteger n, BigInteger m) – функция возвращает обратный элемент от n по модулю m используя расширенный алгоритм Евклида

- BigInteger GenP(BigInteger q, int length, int rm) – функция производит генерацию числа p (модуль эллиптической кривой) из q

- CheckP(BigInteger p, BigInteger q, int length) – фукция проверки условия p^t не равно 1(mod q) для t= 1..31 или t= 1..131

- Point SetPointP() – функция создает начальную точку P

- class Point – содержит переменные a, b, x, y, mod точки эллиптической кривой

- Point(BigInteger x, BigInteger y, BigInteger a, BigInteger b, BigInteger mod) – метод-конструктор для задания точки

- Point operator+(Point p1, Point p2) – оператор сложения двух точек

- Point Double(Point p) – функция удвоения точки

- Point Multiply(Point p, BigInteger n) – функция умножения точки на число

-int GetLen(string s) – функция возвращает длину хэш из параметра командной строки

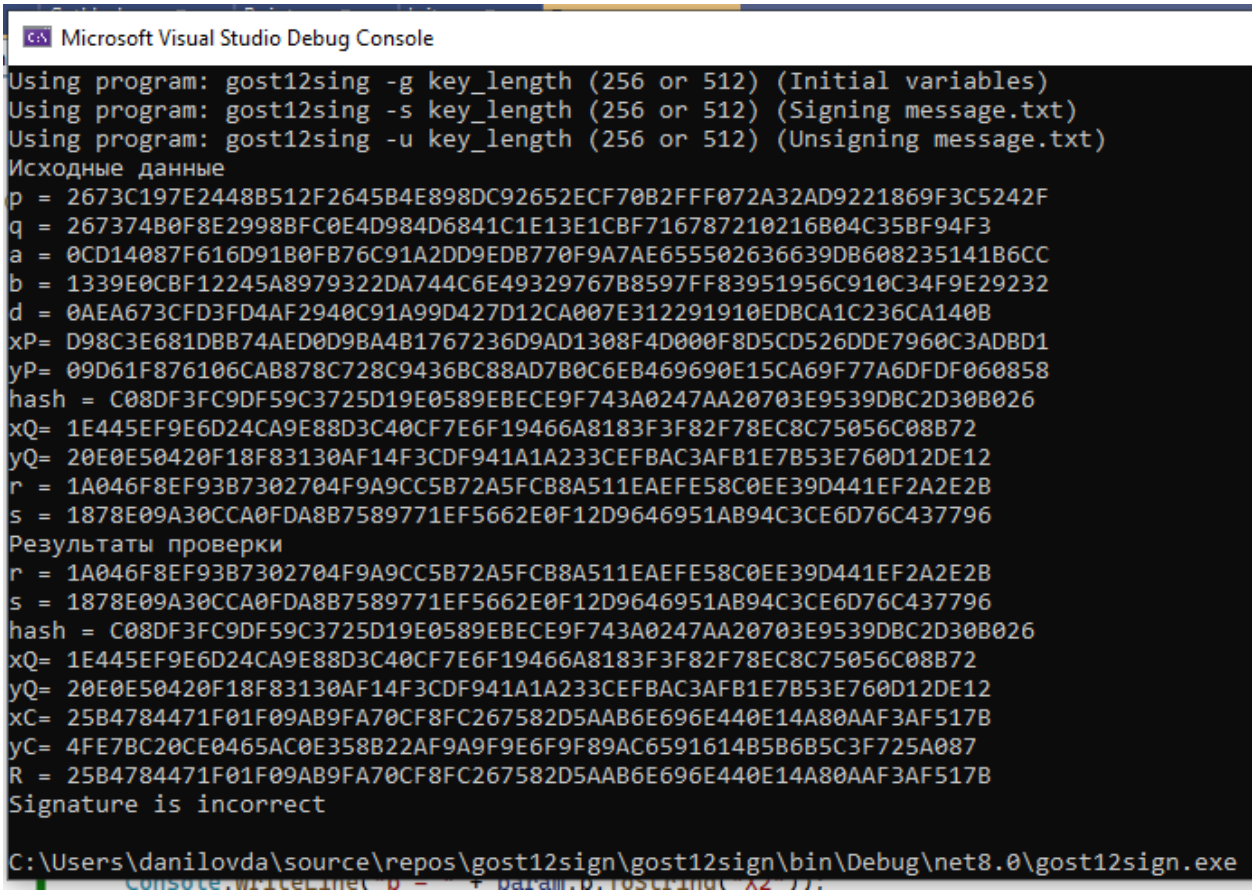
- InitData(int len) – метод запускает первоначальную настройку параметров

- Init ReadData() – функция считывает параметры из файлов

- void Sign(string mfile, Init init, int len) – метод запускает процедуру формирования электронной подписи

- void Check(string mfile, Init init, int len) – метод запускает проверку электронной подписи

6. Результаты работы программы:



5 Выводы о проделанной работе

Были изучены стандарты шифрования ГОСТ 34.11-2012 и ГОСТ 34.10-2012. Была осуществлена программная реализация вычисления хэш функции «Стрибог», формирования электронной цифровой подписи и проверки электронной цифровой подписи. Как подзадачи были реализованы: алгоритм Миллера-Рабина (проверка числа на простоту), вычисление символа Лежандра по критерию Эйлера, извлечение квадратного корня из большого числа, извлечение квадратного корня из большого числа по модулю, функция нахождения обратного элемента по расширенному алгоритму Евклида, функция сложения разных точек эллиптической кривой, функция сложения одинаковых точек эллиптической кривой, функция умножения точек эллиптической кривой на число.

В процессе выполнения задания создать полностью генеративную модель не удалось ввиду сложности генерации точки P эллиптической кривой (в NIST это точка G) в связке с другими параметрами эллиптической кривой. Если же задать часть параметров вручную, использовать числа Мерсенна, использовать уже известные зависимости, то реализация алгоритма не представляет сложности.

6 Список использованных источников

1. https://www.altell.ru/legislation/standards/gost-34.10-2012.pdf
2. https://www.altell.ru/legislation/standards/gost-34.10-2012.pdf
3. https://ru.wikipedia.org/wiki/%D0%93%D0%9E%D0%A1%D0%A2_34.10-2018
4. https://wp.wiki-wiki.ru/wp/index.php/%D0%93%D0%9E%D0%A1%D0%A2_%D0%A0_34.10-2012
5. https://habr.com/ru/articles/335906/