1. Министерство науки и высшего образования Российской Федерации
2. Санкт-Петербургский Политехнический Университет Петра Великого
3. —

**Институт кибербезопасности и защиты информации**

**ЛАБОРАТОРНАЯ РАБОТА №5**

1. «ПРОТОКОЛ ЭЛЕКТРОННОЙ ЦИФРОВОЙ ПОДПИСИ
2. ГОСТ Р 34.10-2018»
3. по дисциплине «Криптографические методы защиты информации»
4. Выполнил
5. студент гр. 4851003/70801 Гасанов Э.А.

<*подпись*>

1. Преподаватель
2. ассистент Ярмак А.В.

<*подпись*>

1. Санкт-Петербург
2. 2021

**Цель**

Изучение протокола электронной цифровой подписи ГОСТ Р 34.10−2018, безопасность которого основана на задаче дискретного логарифмирования в группе точек эллиптической кривой.

**Задача**

1. Вариант задания — 4
2. Разработать программу (**П-1**), реализующую протокол подписи согласно ГОСТ Р 34.10-2018. Параметры криптосистемы соответствуют варианту задания. Описание формата файла подписи представлено в соответствии с Приложением Г. Программа должна поддерживать функции формирования и проверки подписи, а также допускать возможность использования различных ключей.

**Ход работы**

Была написана программа на Python, реализующая схему протокол подписи согласно ГОСТ Р 34.10-2018.

Параметры криптосистемы, полученные согласно варианту залания, представлены в таблице 1.

|  |  |
| --- | --- |
| Параметр | Значение |
| P | 57896044625259982827082014024491516445703215213774687456785671200359045162371 |
| m=r=q | 28948022312629991413541007012245758222850495633896873081323396140811733708403 |
| a | -1 |
| b | 53956679838042162451108292176931772631109916272820066466458395232513766926866 |
| x | 12933162268009944794066590054824622037560826430730236852169234350278155715869 |
| y | 18786030474197088418858017573086015439132081546303111294023901101650919011383 |

Для проверки подписи был использован файл со следующим содержимым (рисунок 1).

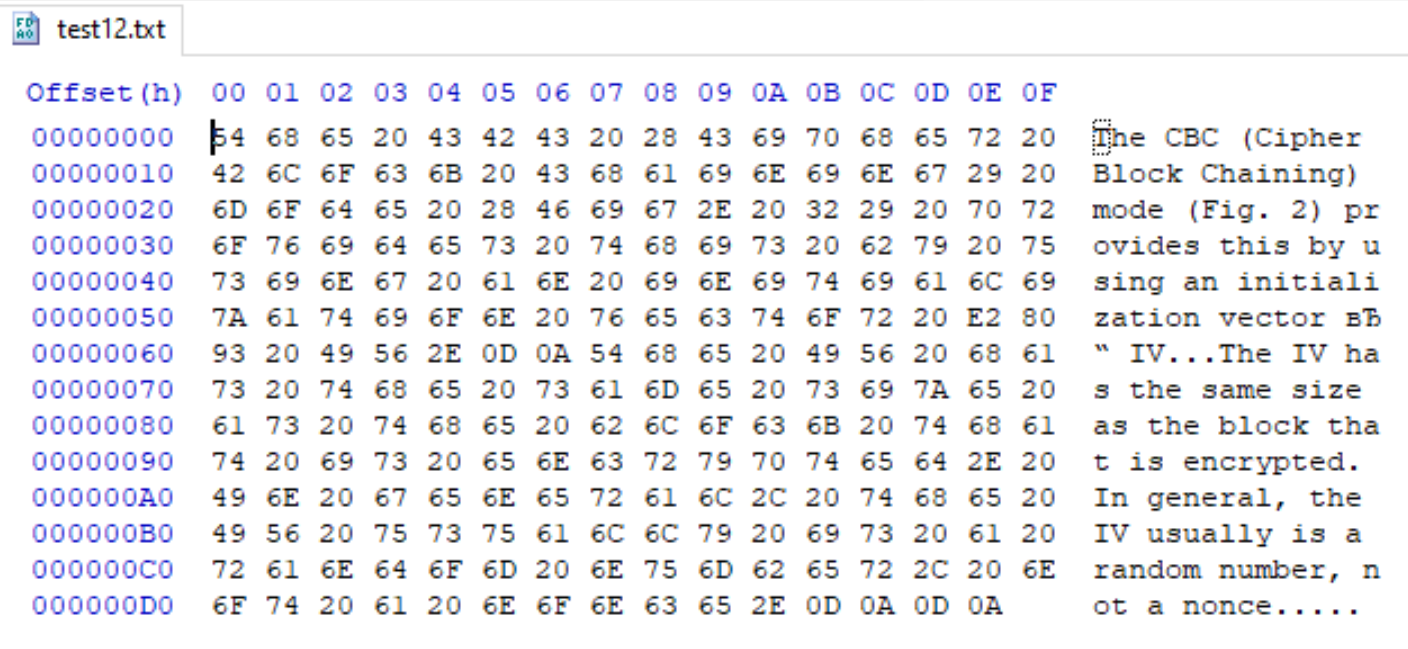


Рисунок 1 – Подписываемый файл.

Хэш-образ всего файла:

0xf467cf40b2fb02df110242ad6b9d52d21530c7c25972fbe240598cffd28547aeL

Файл подписи в формате ASN1 представлен на рисунке 2.

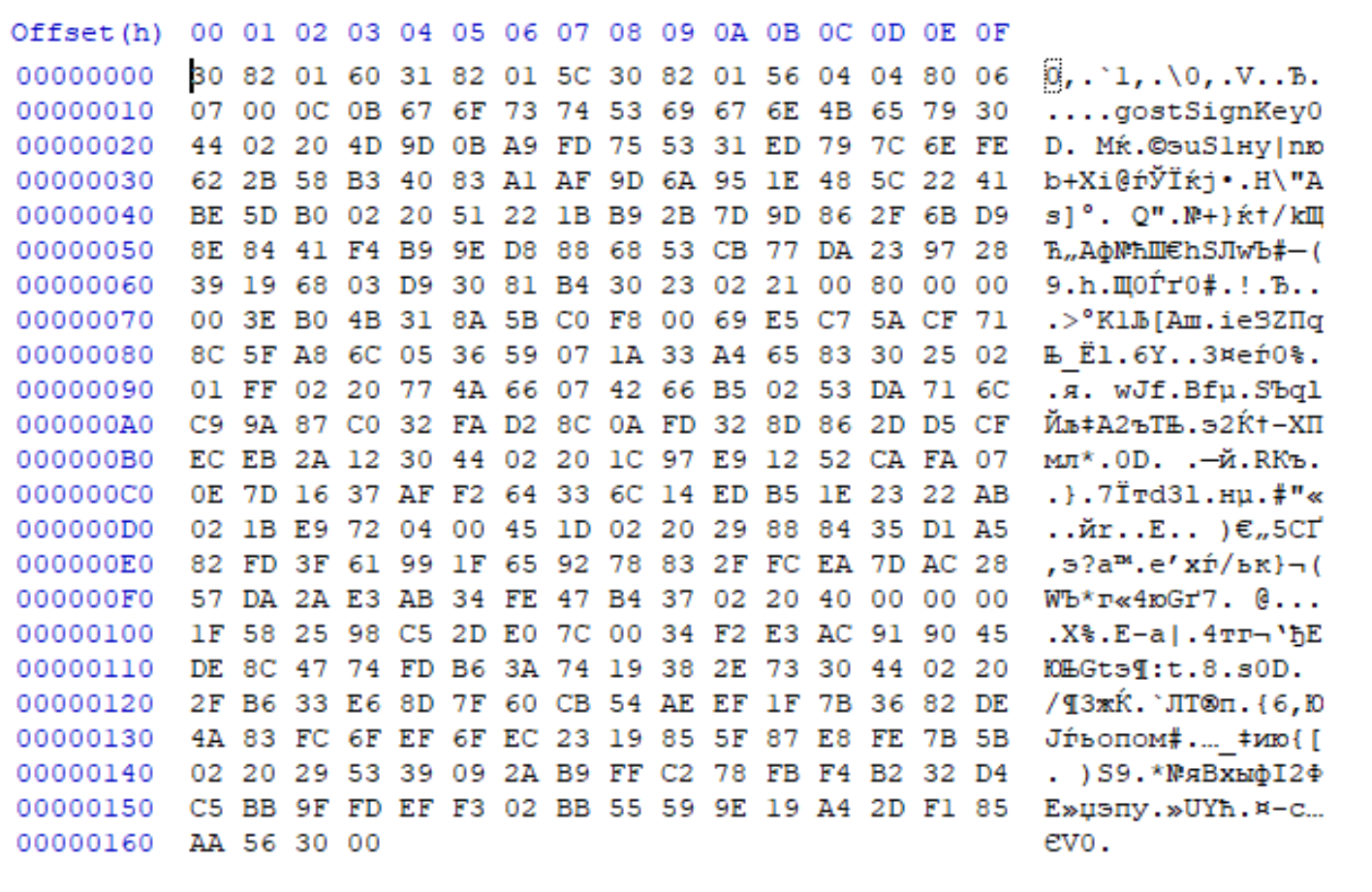


Рисунок 2 – Содержимое файла с подписью.

Подпись в формате ASN1:

0 384: SEQUENCE {

4 380: SET {

8 374: SEQUENCE {

12 4: OCTET STRING 80 06 07 00 // идентификатор алгоритма

18 11: UTF8String 'gostSignKey'

31 68: SEQUENCE { // значение открытого ключа

33 32: INTEGER // координата х точки Q (открытый ключ расшифрования)

: 09 50 DC B8 BB 6D E6 69 F2 95 CA BE 54 97 9C 2A

: BD 50 25 E1 C9 03 2E AE 50 7E F4 71 80 E0 4C 66

67 32: INTEGER // координата y точки Q (открытый ключ расшифрования

: 0B BE FF 79 95 E4 03 81 94 B3 06 F3 21 73 CE 9A

: 57 B9 50 70 E5 30 8D 1E 27 00 53 3F 46 17 CA 3C

: }

101 212: SEQUENCE { // параметры криптосистемы

104 35: SEQUENCE { // параметры поля

106 33: INTEGER // простое число p

: 00 80 00 00 00 61 CF 6C 22 A3 5F 4F 69 AB 7F 03

: 45 58 0D 84 9C B8 BA 68 8E 70 75 5C 32 39 4D DF

: E3

: }

141 69: SEQUENCE { // параметры кривой

143 33: INTEGER // коэффициент а

: 00 80 00 00 00 61 CF 6C 22 A3 5F 4F 69 AB 7F 03

: 45 58 0D 84 9C B8 BA 68 8E 70 75 5C 32 39 4D DF

: E2

178 32: INTEGER // коэффициент b

: 62 6B 67 2F 25 62 73 E0 03 5E 50 75 F1 2F 41 E4

: 9A 2A D8 42 C3 3D 94 C9 60 9E 4D DF 67 49 C8 9A

: }

212 68: SEQUENCE { // образующая группы точке кривой

214 32: INTEGER // x-координата образующей точки P

: 05 23 18 F2 14 E1 63 2F 1D 78 0E 3C 81 81 F0 FB

: 33 19 5E 5E 9F A3 9D 8D C2 45 CB E1 AD 8C E0 02

248 32: INTEGER // y-координата образующей точки P

: 2D EC B4 CF D5 58 7E B0 D6 4E 83 30 2E 5A BF AB

: D6 F8 98 FD 81 1F 2A 32 0B 7B F9 E3 26 8F CA 79

: }

282 32: INTEGER // порядок группы q

: 40 00 00 00 30 E7 B6 11 51 AF A7 B4 D5 BF 81 A2

: AB 17 68 AF 95 71 65 31 02 F7 6D 5C 9F 34 04 89

: }

316 68: SEQUENCE { // подпись сообщения

318 32: INTEGER // число r

: 08 45 19 A2 86 CB 05 96 09 41 D6 A5 59 34 30 01

: 31 18 F2 C1 63 F8 48 33 B2 21 4C F3 DE CC 8E F3

352 32: INTEGER // число s

: 18 C8 F5 71 10 2A EC 2C 2D 4B 8E 47 03 82 F9 74

: 47 C1 FF 14 0D 2D AE D3 1C FD C9 89 2D 9F 75 E6

: }

: }

386 0: SEQUENCE {}

: }

: }

**Контрольная вопросы**

1. **Перечислите преимущества криптосистем на эллиптических кривых по сравнению с другими криптосистемами.**

* Основным преимуществом эллиптической криптографии является малый размер ключа относительно других схем асимметричного шифрования. Это свойство особенно важно при реализации криптографических протоколов в условиях ограниченности ресурсов памяти и производительности, например при программировании смарт-карт. То есть при реализации арифметики на эллиптических кривых для записи чисел, как правило, бывает достаточно 256 разрядов. В результате типовые криптографические операции, такие, как формирование подписи или выработка общего секретного ключа требуют в несколько раз меньших вычислительных затрат, чем при использовании других асимметричных криптоалгоритмов. Эго особенно актуально для мобильных устройств и смарт-карт
* Задача дискретного логарифмирования имеет экспоненциальную сложность, которая при этом практически не снижается со временем.

1. **Почему в стандарте ГОСТ Р 34.10-2018 введено требование ?**

В случае, когда m=p криптосистема уязвима к MOV-атаке, где группа точек эллиптической кривой изоморфна аддитивной группе простого поля. Тогда в этом случае задача дискретного логарифмирования является не сложной. То есть сводит задачу дискретного логарифмирования на эллиптических кривых к задаче дискретного логарифмирования в конечном поле.

1. **Если нарушитель имеет возможность обращать хэш-функцию, как он может подделать сообщение и подпись?**

𝑙∈ℤ,

𝑙(𝑃−𝑄)=(𝑥′,𝑦′),

𝑟≡𝑥′(𝑚𝑜𝑑 𝑞),

𝑠=𝑟,

𝑒≡𝑟∗𝑙−1(𝑚𝑜𝑑 𝑞),

𝑚′≡ℎ−1(𝑒),

где 𝑚′ – поддельное сообщение.

При проверке подписи имеем следующее: 𝑧1𝑃+𝑧2𝑄=𝑠𝑣𝑃+(−𝑟𝑣)𝑄=𝑟𝑒−1𝑃+(−𝑟𝑒−1)𝑄=𝑟𝑒−1(𝑃−𝑄)=(𝑥′,𝑦′)

Если нарушитель имеет возможность обращать хэш-функцию, то он может обратить хэш-значение оригинального сообщения и получить другой прообраз, отличный от оригинала. Так как протокол подписывает хэш-значение, то оба прообраза будут подписаны, и поддельное сообщение можно выдавать за оригинальное.

1. **Почему случайный показатель не должен повторяться в течение времени жизни ключа?**

Потому что по известному можно восстановить секретный показатель следующим образом:

Если на одном и том же ключе d подписывают два сообщения и при этом k повторится, то решая систему из двух уравнений восстанавливается ключ подписи d.

**Вывод**

Был изучен протокол подписи на основе эллиптических кривых. Также были изучены некоторые ограничение на использование параметров, позволяющие обеспечить безопасность протокола.

**Приложение**

Исходный код программы:

from random import randint

from pygost import gost34112012256 as do\_hash

import asn1

def sign(M, P,q, m, d,p,a,b):

h = do\_hash.new(M).digest()[::-1]

h = int(h.encode('hex'), 16)

e = h % q

if e == 0:

e = 1

while True:

k = randint(1,q)

C = k \* P

r = ZZ(C.xy()[0]) % ZZ(q)

if r != 0:

s = (r\*d+k\*e) % q

if s != 0:

break

return r,s

def ASN1encode(P,Q,p,m,a,b,r,s,save\_name):

Qx,Qy=(Q).xy()

Px,Py=P.xy()

encoder = asn1.Encoder()

encoder.start()

encoder.enter(asn1.Numbers.Sequence)

encoder.enter(asn1.Numbers.Set)

encoder.enter(asn1.Numbers.Sequence)

encoder.write(b'\x80\x06\x07\x00', asn1.Numbers.OctetString)

encoder.write('gostSignKey', asn1.Numbers.UTF8String)

encoder.enter(asn1.Numbers.Sequence)

encoder.write(int(Qx), asn1.Numbers.Integer)

encoder.write(int(Qy), asn1.Numbers.Integer)

encoder.leave()

encoder.enter(asn1.Numbers.Sequence)

encoder.enter(asn1.Numbers.Sequence)

encoder.write(p, asn1.Numbers.Integer)

encoder.leave()

encoder.enter(asn1.Numbers.Sequence)

encoder.write(a, asn1.Numbers.Integer)

encoder.write(b, asn1.Numbers.Integer)

encoder.leave()

encoder.enter(asn1.Numbers.Sequence)

encoder.write(int(Px), asn1.Numbers.Integer)

encoder.write(int(Py), asn1.Numbers.Integer)

encoder.leave()

encoder.write(m, asn1.Numbers.Integer)

encoder.leave()

encoder.enter(asn1.Numbers.Sequence)

encoder.write(r, asn1.Numbers.Integer)

encoder.write(s, asn1.Numbers.Integer)

encoder.leave()

encoder.leave()

encoder.enter(asn1.Numbers.Sequence)

encoder.leave()

encoder.leave()

encoder.leave()

encoded\_bytes = encoder.output()

f = open(save\_name, "wb")

f.write(encoded\_bytes)

f.close()

def verify(M, r, s, P, Q):

q = P.order()

if r > q or s > q:

return False

h = do\_hash.new(M).digest()[::-1]

h = int(h.encode('hex'), 16)

print hex(h)

e = h % q

if e == 0:

e = 1

v = (ZZ(e) % q) ^ (-1)

z1 = s\*v % q

z2 = -r \* v % q

C = z1 \* P + z2 \* Q

R = ZZ(C.xy()[0]) % ZZ(q)

return R == r

def ASN1decode(read\_name\_cert):

f = open(read\_name\_cert, "rb")

cert = f.read()

f.close()

decoder = asn1.Decoder()

decoder.start(cert)

decoder.enter()

decoder.enter()

decoder.enter()

t,magic = decoder.read()

t,string = decoder.read()

decoder.enter()

t,Qx = decoder.read()

t,Qy = decoder.read()

decoder.leave()

decoder.enter()

decoder.enter()

t,p = decoder.read()

decoder.leave()

decoder.enter()

t,a = decoder.read()

t,b = decoder.read()

decoder.leave()

decoder.enter()

t,Px = decoder.read()

t,Py = decoder.read()

decoder.leave()

t,m = decoder.read()

decoder.leave()

decoder.enter()

t,r = decoder.read()

t,s = decoder.read()

decoder.leave()

decoder.leave()

decoder.enter()

decoder.leave()

decoder.leave()

decoder.leave()

return Qx,Qy,p,a,b,Px,Py,m,r,s

def verifyASN1(asn\_file, check\_file):

Qx,Qy,p,a,b,Px,Py,m,r,s = ASN1decode(asn\_file)

E = EllipticCurve(FiniteField(p),[a,b])

P = E(Px, Py)

Q = E(Qx, Qy)

f = open(check\_file, "rb")

data = f.read()

f.close()

return verify(data, r, s, P, Q)

def signASN1(P,m,p,q,d,a,b,read\_name, save\_name):

f = open(read\_name, "rb")

M = f.read()

f.close()

r,s = sign(M, P,q, m, d,p,a,b)

ASN1encode(P,P\*d,p,m,a,b,r,s,save\_name)

def main():

print "Start!"

p= 57896044625259982827082014024491516445703215213774687456785671200359045162371

m= 28948022312629991413541007012245758222850495633896873081323396140811733708403

a = -1

b= 53956679838042162451108292176931772631109916272820066466458395232513766926866

x = 12933162268009944794066590054824622037560826430730236852169234350278155715869

y= 18786030474197088418858017573086015439132081546303111294023901101650919011383

E = EllipticCurve(FiniteField(p),[a,b])

P = E(xP, yP)

q = P.order()

print q

d = randint(2,q)

print "Here"

# read\_name = raw\_input("Enter filename to sign\n")

# save\_name = raw\_input("Enter filename where to save signature\n")

read\_name = "С:\\test.txt"

save\_name = "С:\\sign.asn1"

print "Here2"

signASN1(P,m,p,q,d,a,b,read\_name, save\_name)

print verifyASN1(save\_name, read\_name)

main()