DCC IME USP

Routo Terada
Sala 204C Tel.: 3091 5918
e-mail rt@ime.usp.br

MAC 336 - Criptografia e Segurança de Dados
Primeiro Semestre de 2020

Exercício-Programa-2 Data de entrega : veja no paca.ime.usp.br

- Este exercício é para ser feito *individualmente*.
- Entregue no sistema paca.ime.usp.br um ÚNICO arquivo contendo os arquivos seguintes, eventualmente comprimidos:
  - um **único** arquivo com sufixo .PY para execução na linha de comando do Sage
  - um arquivo chamado LEIA.ME (em formato .txt) com:
    * seu nome completo, e número USP,
    * os nomes dos arquivos inclusos com uma breve descrição de cada arquivo,
    * qual computador, e qual versão do SageMath foram usados (modelo, versão, etc..),
- Coloque comentários no seu programa explicando o que cada etapa do programa significa! Isso será levado em conta na sua nota.
- Faça uma saída clara! Isso será levado em conta na sua nota.
- Não deixe para a última hora. Planeje investir 70 porcento do tempo total de dedicação em escrever o seu programa todo ANTES de digitar o programa. Isso economiza muito tempo e energia.
- A nota será diminuida de um ponto a cada dia "corrido" de atraso na entrega.

Objetivo

Este exercício-programa consiste em elaborar um ÚNICO programa executável na linha de comando do SAGE as suas soluções da aplicação dos dois algoritmos de Menezes-Vanstone (página 167 do livro-texto). Deverá ser entregue um **único** arquivo com extensão **.PY** que tenha sido editado em algum editor de texto (.TXT); para poder usar o Sage dentro desse arquivo, ele deve começar com:

```
from sage.all import *.
```

Algoritmos Menezes-Vanstone sobre curva elíptica (página 167 do livro-texto). Dado um primo $p > 2$,

1. O conjunto de legíveis é $Z_p^* \times Z_p^*$ (2 ints.).
2. O conjunto de ilegíveis é $E \times Z_p^* \times Z_p^*$ (4 ints.)
3. A chave *pública* é um par de pontos (Q,P) de E
4. A chave *secreta* é s tal que $Q = sP$
5. Para **criptografar** $X = (x_1,x_2) \in Z_p^* \times Z_p^*$ calcular $Y = (y_0,y_1,y_2)$ da seguinte forma (NONCE):
   a. escolher $k \in Z_{p|}$
   b. $y_0 = kP \in E$
   c. calcular $(c_1,c_2) = kQ$

d. $y_1 = c_1 x_1 \bmod p$

e. $y_2 = c_2 x_2 \bmod p$

---

1. Para **decriptografar** $Y = (y_0, y_1, y_2)$:

a. calcular $s \times y_0 = (c_1, c_2)$

b. $y_1 (c_1)^{-1} \bmod p = x_1$ (pois $y_1 (c_1)^{-1} \bmod p = (c_1 x_1)(c_1)^{-1} = x_1$)

c. $y_2 (c_2)^{-1} \bmod p = x_2$ (pois $y_2 (c_2)^{-1} \bmod p = (c_2 x_2)(c_2)^{-1} = x_2$)

Execução na linha de comando do SAGE

O seu programa, por exemplo chamado EP, deve ser executado na linha de comando do Sage, da seguinte forma:

```
sage EP.py documentoX
```

onde `documentoX` é o nome parâmetro a ser criptografado, decripitografado, etc.

Será **publicado** no paca.ime.usp um arquivo chamado `ep1_esqueleto.py` escrito por Thales Paiva, que poderá servir de ponto de partida para elaborar o seu EP. Os valores numéricos nesse arquivo são *fictícios* e foram sorteados apenas para testes. V pode alterar o arquivo apropriadamente.

O que o seu programa deve fazer

Faça uma saída clara! Isso será levado em conta na sua nota.

1. É dada a curva elíptica $y^2 = x^3 + 2x + 3$ sobre o corpo finito Z^*_{263} ($p = 263$ é primo)

2. Verificar se o ponto $P = (200, 39)$ pertence a essa curva

3. Calcular e mostrar quantos pontos existem nessa curva

4. Calcular e mostrar os primeiros 10 pontos nessa curva: $P, 2P, 3P, \ldots$.

5. Verificar se o ponto $R = (175, 80)$ pertence a essa curva

6. Somar P e R e mostrar o resultado

7. Calcular $s =$(seu NUSP)$\bmod 263$. Por exemplo: $52713549 \bmod 263 = 196$. Se resultar $s = 0$, some 1 sucessivamente ao seu NUSP e tente várias vezes até resultar $s \neq 0$. Na realidade s deveria ser gerado aleatoriamente e ser armazenado secretamente, mas para exercitar vamos admitir que não seja.

8. Calcular $sP = Q$ e mostrar Q. Chave ´pública da Alice é (Q, P)

9. Criptografar R e mostrar o resultado $Y = (y_0, y_1, y_2)$

10. Decriptografar $Y = (y_0, y_1, y_2)$ e mostrar o resultado

11. Criar e mostrar (pelo menos os primeiros 100 bytes) um `documento1` de pelo menos 100K bytes, formado pelo seu número USP, NUSP, concatenado com letras (bytes) geradas através do uso de um gerador de números pseudo-aleatórios. Por exemplo, algo como $\underbrace{52713549}4a1ffa82bc88\ldots 9abb$ se for em notação hexadecimal.

12. Criptografar `documento1` em modo CBC (Cipher Block Chaining, pg. 114 do livro-texto) com Valor Inicial - VI - igual a bits $000\ldots0$, e mostrar (pelo menos os primeiros 100 bytes) o resultado, `doc1-cript`

13. Decriptografar `doc1-cript` e mostrar o resultado `doc1-cript-inverso`

14. Mostrar a distância de Hamming entre `doc1-cript` e `doc1-cript-inverso`. Qual o resultado desejável: distância grande ou pequena? Justifique.

15. Criar `documento2` igual ao `documento1` exceto que o seu NUSP é acrescido de 1 (inteiro) no PRIMEIRO dígito; as letras geradas não são alteradas: $\underbrace{627135494}a1ffa82bc88\ldots 9abb$.

16. Criptografar `documento2` em modo CBC e mostrar (pelo menos os primeiros 100 bytes) o resultado, `doc2-cript`

17. Calcular e mostrar a distância de Hamming entre `doc1-cript` e `doc2-cript`. Qual o resultado desejável: distância grande ou pequena? Justifique.

18. Gerar um outro segredo para Beto: s_B tal que $Q_B = s_B P$, com o mesmo P usado antes para a Alice.

19. Criptografar com a chave do Beto o mesmo `documento1` gerado anteriomente, em modo CBC e mostrar (pelo menos os primeiros 100 bytes) o resultado, `doc1-cript-Beto`

20. Calcular e mostrar a distância de Hamming entre `doc1-cript` e `doc1-cript-Beto`. Qual o resultado desejável: distância grande ou pequena? Justifique.

———————— FIM ————————