DCC IME USP

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

MAC336-5723 - Criptografia e Segurança de Dados
Primeiro Semestre de 2022

Exercício-Programa Data de entrega : veja no e-disciplinas/USP da disciplina

- Este exercício é para ser feito *individualmente*.
- **Este exercício deve ser resolvido em Python3.**
- Entregue no sistema e-disciplinas um **ÚNICO** arquivo comprimido (zip, tar.gz) contendo os arquivos seguintes comprimidos:
  - um **único** arquivo chamado ep1.py com a solução do EP.
  - um arquivo chamado **LEIA.ME** (em formato TXT ou Markdown) 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 Python3 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 diminuída 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 com os dois esquemas criptográficos descritos abaixo. 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 *.
```

Exercício-programa Algorimos de assinatura e verificação Menezes-Vanstone com curva elíptica

É dada a curva elíptica $y^2 = x^3 - 7x$ sobre o corpo finito Z_{271}^* ($q = 271$ é primo).

Algoritmo para assinar $x : 0 < x < n$

Entrada: inteiro primo $q > 0$, curva elíptica irredutível $y^2 = x^3 + ax + b$ sobre Z_q^*,

$\boxed{n = ordem(P)}$

$x : 0 < x < n$, $Q = sP$, s é a chave privada secreta, (Q,P) é a chave pública da Alice

Saída: assinatura (r,a)

1- Escolher k no intervalo $[1, n-1]$ (i.e., NONCE)

2- Calcular ponto $kP = (x_1, x_2)$ e $r = x_1 \bmod n$, x_1, x_2 no intervalo $[0, n-1]$ Se $r = 0$, repetir o Passo 1

3- Calcular $k^{-1} \bmod n$
4- $a = k^{-1}[hash(x) + s \times r] \bmod n$ Se $a = 0$, repete os Passos 1 a 4
5. Assinatura sobre x é (r, a)

Verificação da assinatura (r, a), sobre x.
Entrada: $x, (a, r)$
1- Verificar se r e a são do intervalo $[1, n-1]$. Se não for, rejeitar a assinatura.
2- Calcular $w = a^{-1} \bmod n$
3- Calcular $u_1 = w[hash(x)] \bmod n$, $u_2 = wr \bmod n$
4- Calcula $[u_1 P + u_2 Q] = (x_o, y_o)$ e $v = x_0 \bmod n$
5- Aceita a assinatura se e somente se $v = x_0 = r \bmod n$

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 do arquivo a ser lido pelo seu programa, que esteja eventualmente gravado no mesmo diretório que o EP.py

Será **publicado** no e-disciplinas um arquivo chamado `ep1_esqueleto.py` , 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 $y^2 = x^3 - 7x$ sobre o corpo finito Z^*_{271} ($q = 271$ é primo)
2. Verificar se o ponto $P = (201, 247)$ pertence a essa curva e se gera todos os seus pontos. E calcular quantos pontos são. Mostrar os resultados.
3. Verificar se o ponto $R = (177, 147)$ pertence a essa curva.
4. Somar P e R e mostrar o resultado
5. Calcular e mostrar $s =$(seu NUSP)$\bmod 271$. Por exemplo: $633713549 \bmod 271 = 103$. Se resultar $s = 0$, some 1 sucessivamente ao seu NUSP e tente várias vezes até resultar $s \neq 0$. (Na realidade o segredo s deveria ser gerado aleatoriamente e ser armazenada de forma segura, mas para este exercício vamos supor que seja gerado dessa forma.)
6. Calcular e mostrar $sP = Q$ e mostrar Q
7. Assinar $x = 214$ com o algoritmo dado, resultando a assinatura (r, a) e mostrar.
8. Verificar se (r, a) é a assinatura verdadeira com o algoritmo dado, mostrar o resultado.
9. Criar e mostrar (pelo menos os primeiros 100 bytes de) 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 $\underbrace{527135494}a1ffa82bc88\ldots 9abb$ se for em notação hexadecimal.
10. Criar e mostrar (pelo menos os primeiros 100 bytes) um `documento2` igual ao `documento1` exceto que o seu NUSP é **acrescido de 1** (inteiro); as letras geradas

não são alteradas: 527135495*a*1*ffa*82*bc*88…. 9*abb*. (ou seja, 527135494 + 1 = 527135495)

11. Criar e mostrar:
- **a.** `hash1` aplicando o algoritmo SHA512 sobre `documento1`,
- **b.** e `hash2` aplicando o algoritmo SHA512 sobre `documento2`

12. Fazer Alice assinar esses dois hashings pelo algoritmo Menezes-Vanstone, resultando `assinaturaHash1` e `assinaturaHash2` e mostrar essas assinaturas

13. Calcular e mostrar a distância de Hamming dessas duas assinaturas. O objetivo deste item é o de verificar se uma *pequena* alteração no `documento1` (i.e., entropia baixa) causa, ou não, uma alteração significativa (entropia alta) na assinatura, em posições de bits pseudo-aleatórios. Pergunta: é desejável que seja alta? Por quê?

14. Verificar a validade da `assinaturaHash1` sobre o `documento1` com a chave pública da Alice, e mostrar porquê é válido.

15. Verificar a falsidade da `assinaturaHash1` sobre o `documento2` com a chave pública da Alice, e mostrar porquê é falso.
- **a.** (1) Ler um arquivo texto `arq1.txt`, que será **publicado** no e-disciplinas, aplicar SHA512 sobre `arq1.txt`, e mostrar o resultado `hashA`,
- **b.** (2) gerar com a chave da Alice a assinatura sobre `hashA`, resultando `assinaturaHashA`, e mostrar essa assinatura,
- **c.** (3) verificar e mostrar a autenticidade dessa assinatura `assinaturaHashA`.
- **d.** (4) Alterar uma **única** letra em qualquer posição desse arquivo `arq1.txt`, (Se esse arquivo fosse uma ordem de pagamento por e-mail, essa alteração poderia ser no valor em R$)
- **e.** (5) fazer Alice assinar o resultado dessa alteração e mostrar a assinatura,
- **f.** (6) verificar que a assinatura da Alice é outra, calculando e mostrando a distância de Hamming entre as duas assinaturas.

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

17. Simular duas assinaturas: tanto da Alice como do Beto, da seguinte maneira:
- **a.** gerar e mostrar um outro texto **concatenando** o arquivo texto `arq1.txt` anterior com a assinatura `assinaturaHashA` da Alice, resultando um texto `arq2.txt←arq1.txt|assinaturaHashA` (isto é, `arq2` é `arq1` concatenado com `assinaturaHashA`),
- **b.** mostrar o resultado da aplicação do SHA512 sobre esse `arq2.txt` e
- **c.** fazer Beto assinar esse hash, resultando a assinatura `assinaturaHashB` do Beto depois da Alice ter assinado o documento original `arq1.txt`. Mostrar esse resultado.
- **d.** Verificar e mostrar a autenticidade dessa assinatura `assinaturaHashB` do Beto.

———————— FIM ————————