ECDSA

April 5, 2019

1 ECDSA - *Elliptic Curve Digital Signature Algorithm*

Segue-se a implementação do ECDSA, usando a curva elitica prima **P-521**, definida no **FIPS186-4**. Note-se que, para a mesma, tomou-se como referência as páginas 394 e 395 do seguinte livro: *S. Yan, Number Theory for Computing, Springer, 2002*.

Seja **E** uma curva elíptica sobre $\mathbb{F}_p$, sendo p primo. Seja ainda P um ponto de ordem prima q em $E(\mathbb{F}_p)$. Suponha que Alice deseje enviar uma mensagem assinada para Bob.

1.1 SETUP

A Alice procede da seguinte forma:

1. Selecciona um inteiro aleatório $x \in [1, q-1]$;
2. Calcula $Q = xP$;
3. Torna Q público e mantém x privado.

Desta forma, aAlice gera a chave pública Q e a chave privada x.

1.2 GERAÇÃO DA ASSINATURA

Para assinar a mensagem m, a Alice faz o seguinte:

1. Seleciona novamente um inteiro aleatório, $k \in [1, q-1]$;
2. Determina $kP = (x_1, y_1)$ e $r \equiv x_1 (mod\ q)$. Se $r = 0$, regressa ao ponto 1;
3. Calcula $k^{-1} mod\ q$;
4. Efectua $s \equiv k^{-1}(H(m) + xr)(mod q)$, sendo $H(m)$ o valor *hash* da mensagem. Se $s = 0$, retorna ao passo 1.

A assinatura da mensagem m é o par de inteiros (r, s).

1.3 VERIFICAÇÃO DA ASSINATURA

Para verificar a assinatura, (r, s), da mensagem m da Alice, o Bob deve: 1. Obter um cópia autêntica da chave pública da Alice, Q.

2. Verificar que (r, s) são inteiros compreendidos no intervalo $[1, q-1]$, calcular $kP = (x_1, y_1)$ e $r \equiv x_1 (mod\ q)$;
3. Determinar $w \equiv s^{-1} (mod\ q)$ e $H(m)$;
4. Calcular $u_1 \equiv H(m)\ w\ (mod\ q)$ e $u_2 \equiv r\ w\ (mod\ q)$;
5. Determinar $u_1 P + u_2 Q = (x_0, y_0)$ e $v \equiv x_0 (mod q)$;
6. A assinatura é validade se e só se: $v = r$.

```
In [37]: class ECDSA:
             def __init__(self):
                 p = 2^521 - 1;
                 F = GF(p)
                 A = p - 3
                 B = 1093849038073734274511112390766805569936207598951683748994586394495953116150735016013708737573759623248592132296706313309438452531591012912142327488478985984
                 self.q = 6864797660130609714981900799081393217269435300143305409394463459185543183397655394245057746333217197532963996371363321113864768612440380340372808892707005449
                 E = EllipticCurve([F(A), F(B)])
                 self.P = E.random_point()
                 self.chv_priv = ZZ.random_element(1,self.q-1)
                 self.chv_pub = self.chv_priv*self.P

             def sign(self,m):
                 k = ZZ.random_element(1,self.q-1)
                 x_1 = (k*self.P)[0]
                 y_1 = (k*self.P)[1]
                 r = mod(x_1,self.q)
                 if r ==0:
                     k = ZZ.random_element(1,self.q-1)
                     x_1 = (k*self.P)[0]
                     y_1 = (k*self.P)[1]
                     r = mod(x_1,self.q)
                 k_1 = mod(k^(-1),self.q)
                 h = hash(m)
                 s = mod((k_1*(h+self.chv_priv*r)),self.q)
                 if s ==0:
                     k = ZZ.random_element(1,self.q-1)
                     x_1 = (k*self.P)[0]
                     y_1 = (k*self.P)[1]
                     r = mod(x_1,self.q)
```

```
                if r ==0:
                    k = ZZ.random_element(1,self.q-1)
                    x_1 = (k*self.P)[0]
                    y_1 = (k*self.P)[1]
                    r = mod(x_1,self.q)
                k_1 = mod(k^(-1),self.q)
                h = hash(m)
                s = mod((k_1*(h+self.chv_priv*r)),self.q)
            sig = (r,s)
            return sig

        def verify(self,sig,m):
            r = sig[0]
            s = sig[1]
            w = mod(s^(-1),self.q)
            h = hash(m)
            u_1 = mod(h*w,self.q)
            u_2 = mod(r*w,self.q)
            x_0 = (ZZ(u_1)*self.P+ZZ(u_2)*self.chv_pub)[0]
            v = mod(x_0,self.q)
            return (v == r)
```

In [38]:
```
ECDSA = ECDSA()
msg = "Mensagem";
assinatura = ECDSA.sign(msg)
```

In [39]:
```
ECDSA.verify(assinatura,msg)
```

Out[39]:
```
True
```