TP6

May 30, 2020

1 Teoria de Números Computacionais

1.1 Trabalho 6: Assinatura cega de Chaum

Hugo Sousa (a76257 - LCC)

Matias Capitão (a82726 - LCC)

Rafael Antunes (a77457 - LCC)

1.1.1 Introdução

A assinatura cega, apresentada por David Chaum, é uma forma de assinatura digital na qual o conteúdo de uma mensagem é disfarçado ("blinded") antes de ser assinada. A assinatura "blind" resultante pode ser verificada publicamente em relação à mensagem original "unblinded". Assinaturas cegas normalmente são utilizadas em protocolos relacionados à privacidade, nos quais o assinante e o autor da mensagem são partes diferentes. Exemplos incluem sistemas eleitorais criptográficos e esquemas de dinheiro digital.

1.2 Esquema de assinatura cega de Chaum

Como analogia, consideremos que Alice tem uma carta que deve ser assinada por uma autoridade (por exemplo Bob), mas Alice não deseja revelar o conteúdo da carta a Bob. Ela pode colocar a carta em um envelope, forrado em papel de carbono, e enviá-lo a Bob. Bob assinará a parte externa do envelope sem abri-lo e vai enviar de volta para Alice. Alice pode abri-lo para encontrar a carta assinada por Bob, mas sem Bob ter visto seu conteúdo.

.Um dos esquemas de assinatura cega mais simples é um esquema baseado no RSA, adaptado para assinaturas cegas.

1.2.1 Descrição

No protocolo descrito abaixo assumimos que Bob gerou as suas chaves pública (e, n) e privada (d, n), e tem a função de assinatura tal e qual no esquema de assinatura digital RSA: $S_b(m) = m^d(modn)$.

Assim sendo são feitos os seguintes passos:

- **Passo inicial -** Alice obtém a chave pública (e, n) de Bob e escolhe um número k aleatório tal que $0 < k < n$ e tal que $(k, n) = 1$

- **"Blinding" -** Alice de seguida computa $m' = mk^e (mod n)$ e manda m' a Bob (a $k^e (mod n)$ chamamos o "blinding factor")
- **Assinatura -** Bob computa $s' = m'^d (mod n)$ e manda de volta a Alice
- **"Unblinding" -** Alice computa $s = s' * k^{-1} (mod n)$ e obtém o valor da assinatura válida de Bob $S_b(m)$

Prova Ora este último passo "unblinding" devolve-nos a assinatura de Bob pois:
$s \equiv s' \cdot k^{-1} \equiv (m')^d k^{-1} \equiv m^d k^{ed} k^{-1} \equiv m^d k k^{-1} \equiv m^d \pmod{N}$

1.2.2 Codificação e Exemplo

```
In [164]: #Classe que define um Assinante
          class RSA_Blind_Signer:

              def __init__(self):
                  n,e,d = self.gen_keys()
                  self.n = n # p*q
                  self.public = e # exponent of public key
                  self.private = d # exponent of private key

              def publish(self):
                  return(self.public, self.n)

              #Sign a message blind
              def blind_sign(self, blind):
                  s = power_mod(blind, self.private, self.n)
                  return s

              #Sign a message
              def sign(self, msg):
                  return power_mod(msg, self.private, self.n)

              #Function to generate the public and private keys
              @staticmethod
              def gen_keys(nbits = 32):
                  p = random_prime(2^(nbits//2), 2^(nbits//2-2))
                  q = random_prime(2^(nbits//2 +1), 2^(nbits//2 -1))

                  while p == q:
                      p = random_prime(2^(nbits//2), 2^(nbits//2-2))
                      q = random_prime(2^(nbits//2 +1), 2^(nbits//2 -1))

                  n = p*q
                  m = (p-1)*(q-1)
                  e = randint(2, m-1)
                  while gcd(e, m) != 1:
                      e = randint(2,m-1)
```

```
            d = power_mod(e, -1, m)

            return n,e,d

        #Classe que define um autor de mensagem
        class User:

            def __init__(self, msg):
                self.msg = msg
                self.k = 0

            def blinding(self, pubk):
                k = randint(0, pubk[1])
                while(gcd(k, pubk[1])!= 1):
                    k = randint(0, pubk[1])
                self.k = k
                b_factor = power_mod(k, pubk[0], pubk[1])
                m = mod((self.msg*b_factor), pubk[1])
                return m

            def unblinding(self, s, pubk):
                unblinding_factor= power_mod(self.k, -1, pubk[1])
                signature = mod(s*unblinding_factor, pubk[1])
                return signature
```

In [165]:
```
Bob = RSA_Blind_Signer()

m= 1234

Alice=User(m)
```

In [166]:
```
'''Bob cria e publica a sua chave publica'''
pubk=Bob.publish()
Bob.publish()
```

Out[166]: (466661761, 3287446447)

In [167]:
```
'''Alice procede com o passo "blinding" onde vai multiplicar a mensagem pelo blinding
a Bob'''
b = Alice.blinding(pubk)
Alice.k
```

Out[167]: 451395328

In [168]:
```
'''Bob proce à assinatura cega da mensagem sem ver o seu conteúdo'''
s = Bob.blind_sign(b)
Bob.blind_sign(b)
```

Out[168]: 2730601200

```
In [169]: '''Alice procede ao passo "unblinding" onde obtém a assinatura pretendida'''
          Alice.unblinding(s, pubk)

Out[169]: 2911528931

In [170]: '''Como podemos ver o resultado do Unblinding é igual à assinatura direta da mensagem'
          Bob.sign(1234) == Alice.unblinding(s, pubk)

Out[170]: True
```

1.2.3 Bibliografia

https://en.wikipedia.org/wiki/Blind_signature Wikipedia - Blind Signature

https://www.geeksforgeeks.org/chaumian-blinding/ Geeksforgeeks - Chaumian Blinding

David R. Kohel, Cryptography (sec 9.3)


```
In [ ]:
```