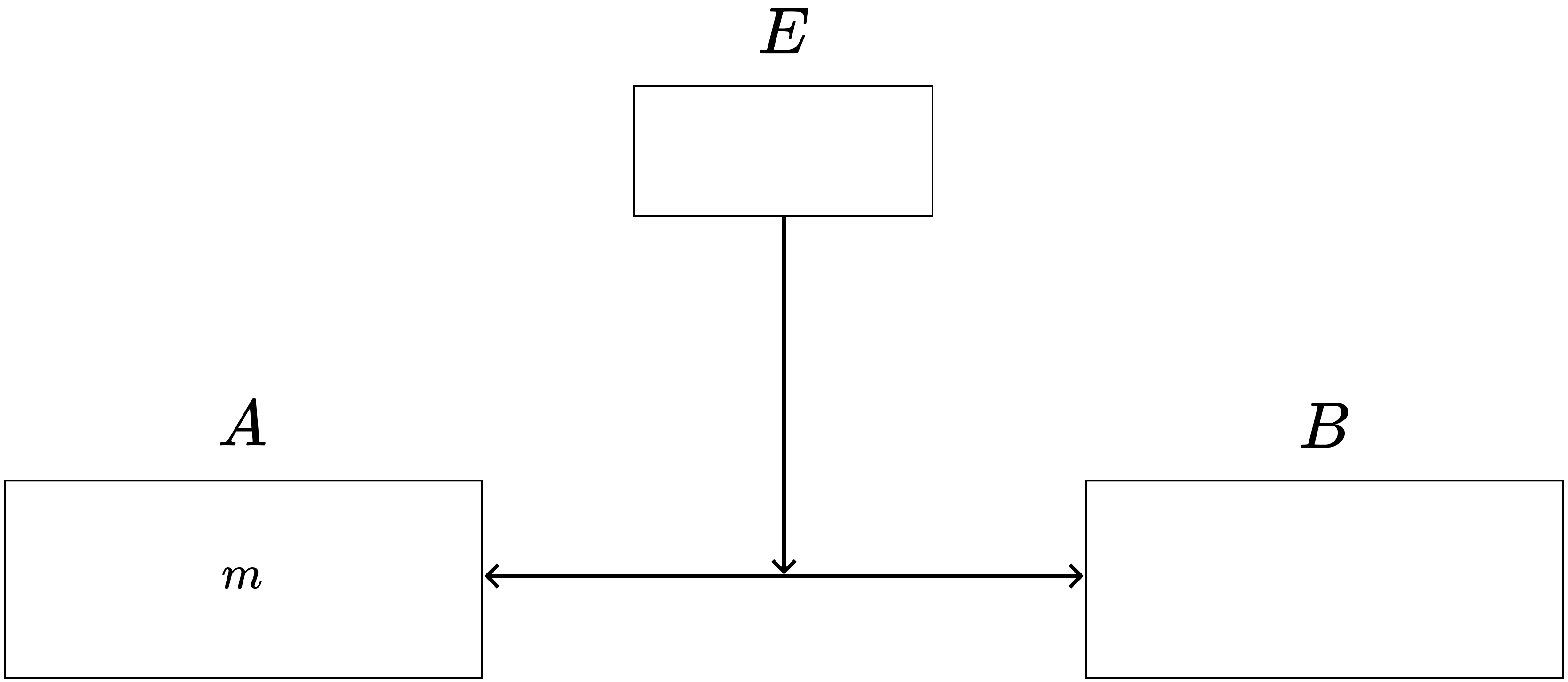
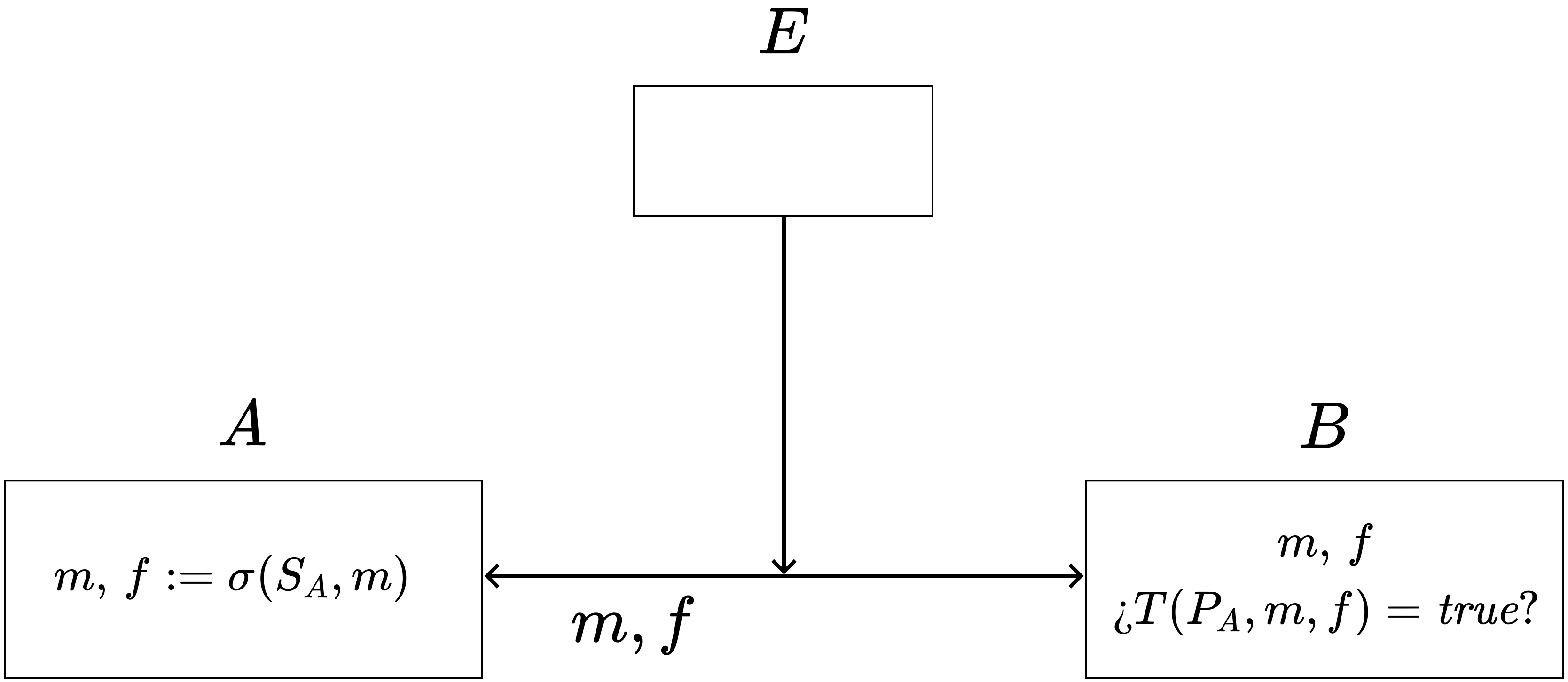
IIC3253

Firmas digitales

Firma digital con una clave pública



Firma digital con una clave pública



Firma digital con una clave pública

- A está firmando un mensaje m, para cualquiera que lo necesite
- $\sigma(S_A, m)$ utiliza la clave secreta de A para generar una firma f de m, de manera tal que solo A puede firmar
- $T(P_A, m, f)$ verifica si f es una firma válida del mensaje m por el usuario A
- $T(P_A, m, f)$ utiliza la clave pública de A, de manera que cualquiera puede verificar si f es una firma válida

Firmas digitales con RSA

Suponga que $P_A = (e, N)$ y $S_A = (d, N)$ son las claves pública y privada de un usuario A

Para cada $m \in \{0, \ldots, N-1\}$, sabemos que

$$Dec(S_A, Enc(P_A, m)) = m$$

Firmas digitales con RSA

Pero también tenemos que

$Enc(P_A, Dec(S_A, m)) =$

Firmas digitales con RSA

Pero también tenemos que

$$\begin{aligned}
Enc(P_A, Dec(S_A, m)) &= (m^d \bmod N)^e \bmod N \\
&= (m^d)^e \bmod N \\
&= m^{d \cdot e} \bmod N \\
&= (m^e)^d \bmod N \\
&= (m^e \bmod N)^d \bmod N \\
&= Dec(S_A, Enc(P_A, m)) \\
&= m
\end{aligned}$$

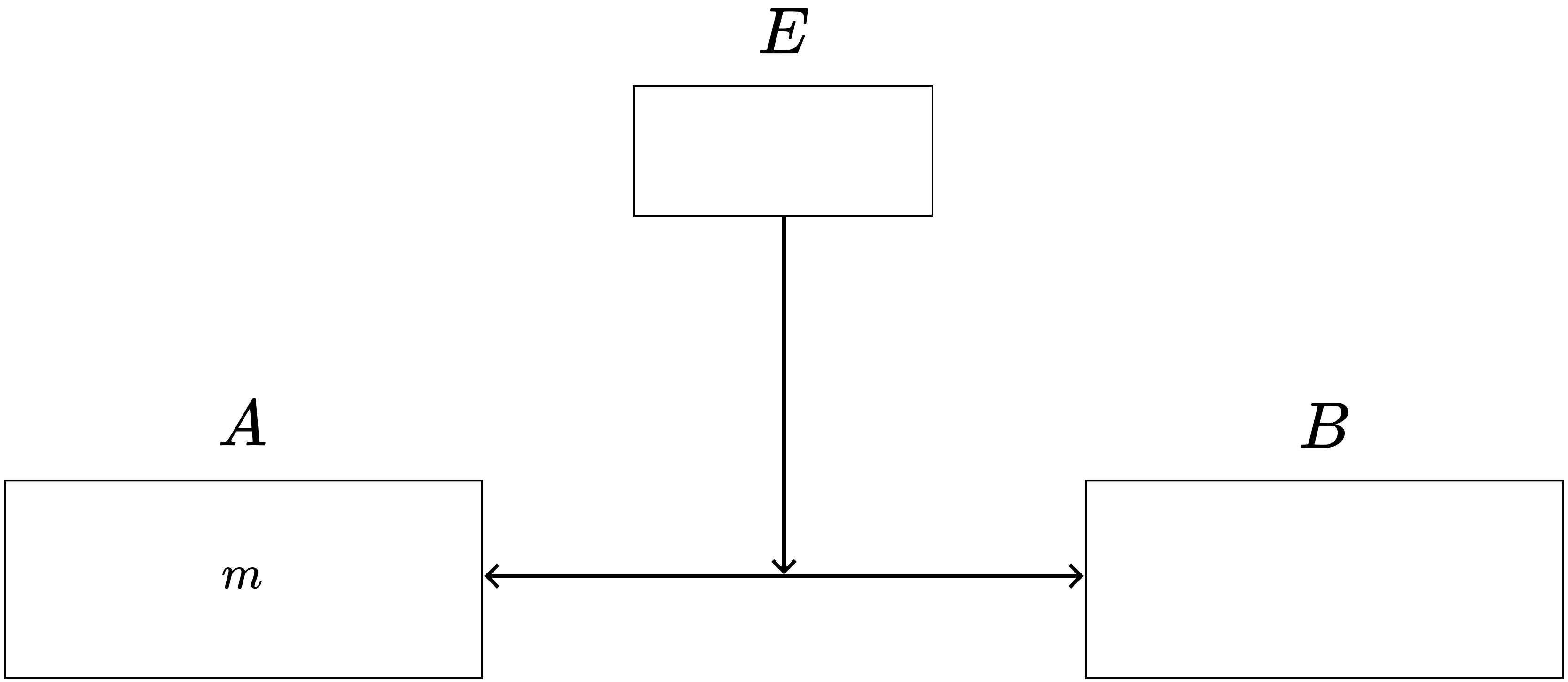
Firmas digitales con RSA

Definimos entonces la firma del mensaje m por el usuario A como:

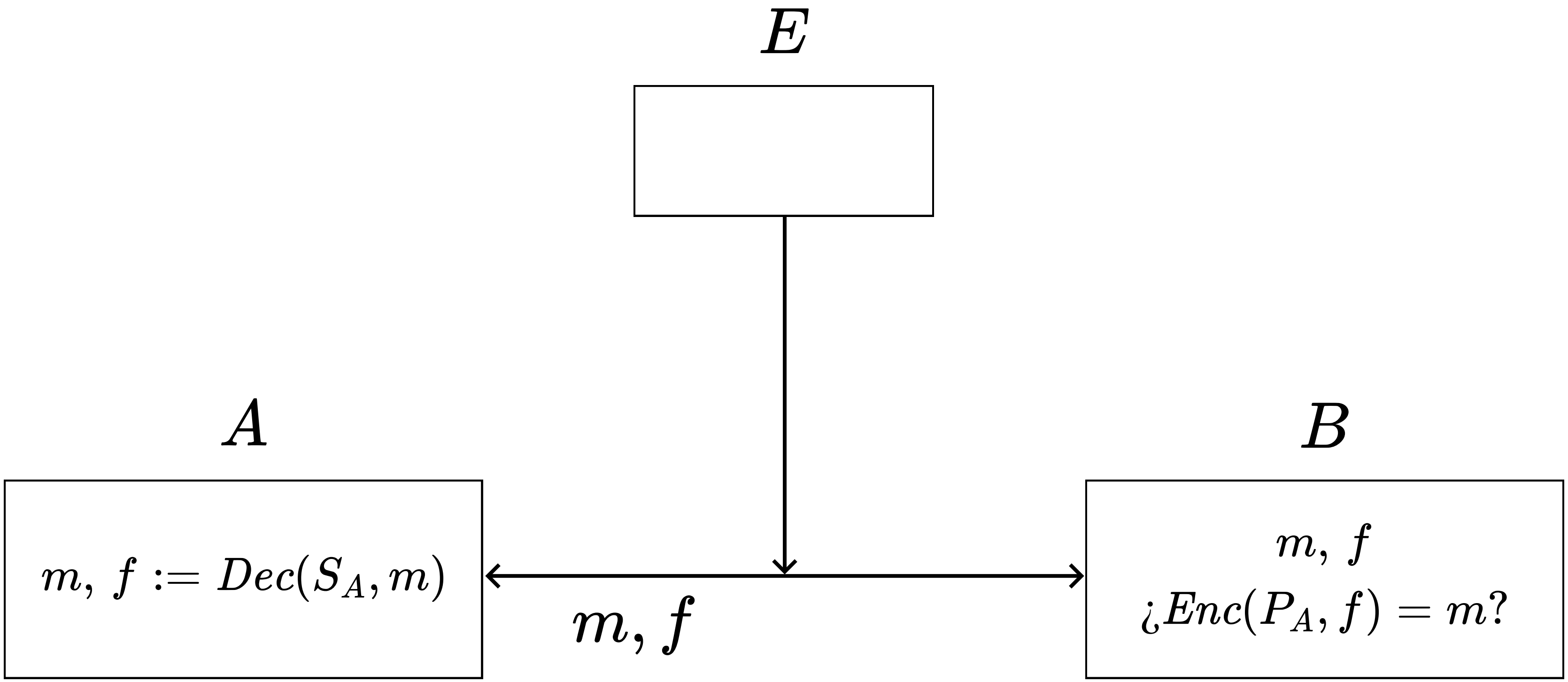
$$f := Dec(S_A, m)$$

Solo A puede generar esta firma. Cualquier usuario puede verificar si A firmó un mensaje usando la clave pública P_A de A

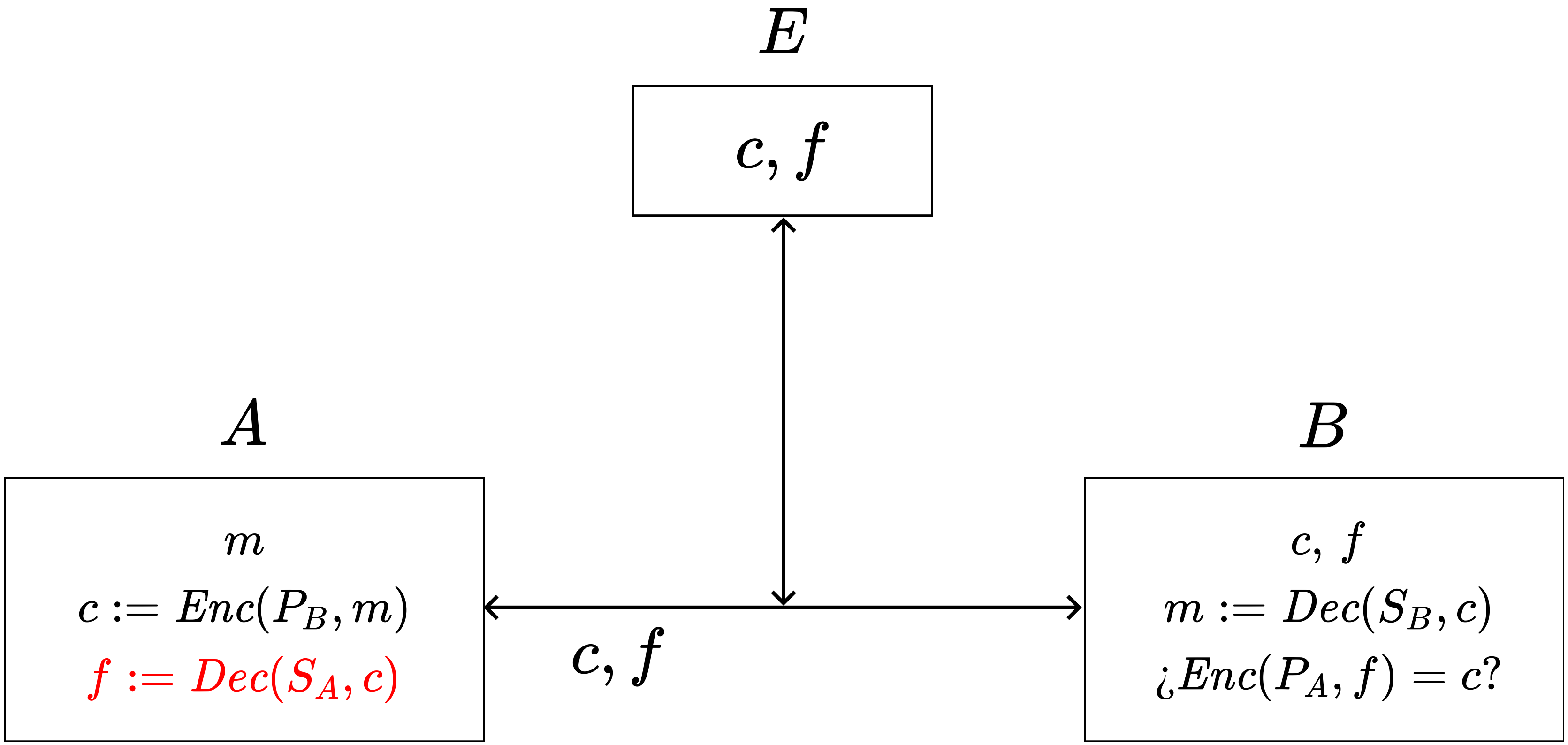
El esquema de firmas digitales con RSA



El esquema de firmas digitales con RSA



A puede firmar para B



¿Qué problema tiene el esquema anterior?

Firmar un mensaje m puede ser lento si m es un mensaje muy largo

Para solucionar este problema, se puede firmar $h(m)$ en lugar de firmar m, donde h es una función de hash

Firmas de Schnorr

Vamos a ver un segundo esquema para firmas digitales que está basado en el problema del logaritmo discreto

Se puede aplicar en cualquier grupo donde el problema de calcular el logaritmo discreto es difícil

La definición de las firmas de Schnorr

Suponemos dado un grupo finito $(G, *)$ y un elemento $g \in G$ tal que $|\langle g \rangle| = q$

- G, g y q son públicos
- Como vimos antes, se debe tener que $|G|$ y q son números grandes

Además suponemos dada una función de hash h

La definición de las firmas de Schnorr

La clave privada de un usuario A es $x \in \{1, \ldots, q - 1\}$ y su clave pública es g^x

La definición de las firmas de Schnorr

La clave privada de un usuario A es $x \in \{1, \ldots, q - 1\}$ y su clave pública es g^x

El usuario A quiere firmar un mensaje m

La definición de las firmas de Schnorr

A firma m de la siguiente forma:

1. Genera al azar $k \in \{1, \ldots, q-1\}$ y calcula $r = g^k$
2. Calcula $v = h(r \| m)$ usando r como un string
3. Calcula $s = k + v \cdot x$ interpretando v como un número natural
4. La firma de m es (v, s)

La verificación de una firma de Schnorr

Se puede verificar que (v, s) es una firma de m generada por A de la siguiente forma:

1. Calcule $\alpha = g^s = g^{k+v\cdot x} = g^k * g^{v\cdot x} = g^k * (g^x)^v$
2. Calcule $\beta = \alpha * ((g^x)^v)^{-1} = g^k * (g^x)^v * ((g^x)^v)^{-1} = g^k$
3. Verifique si $v = h(\beta \| m) = h(g^k \| m)$

Un ejemplo concreto: $\mathbb{Z}_p^*$ y SHA-256

Vamos a ver cómo se calculan las firmas de Schnorr considerando el grupo $(\mathbb{Z}_p^*, \cdot)$ y la función de hash SHA-256

El archivo grupo.txt

1712545831761413793019604197925757782640883232403750857339329298 16

4266713974762177880243877523872859296834461358937993234847561350 34

7693216316697381321869834381646328914418536291260252254049498309 05

3149723296582953652450726984882565831142029933592229570974326750 83

2252596677395039491925757684203877163274204414247105350985012360 58

8381585716266691777519349615737265619555830572700989127600651400 04

0936587721817138831992389630937779176259061431184964296138022485 19

4046042171044936892725297487039587393638790967227488329537748100 81

5047587859027059179835056348816808092380461182238752019805400299 06

23911454389104774092183

El archivo grupo.txt

8041367327046189302693984665026706374844608289874374425728797669509435881459140662650215832833471328470334064628508692231999401840332046192569287351991689963279656892562484773278584208040987631569628520464069532361274047374444344996651832979378318849943741662110395995778429270819222431610927356005913836932462099770076239554042855287138026806960470277326229482818003962004453764400995790974042663675692120758726145869061236443893509136147942414445551848162391468541444355707785697825741856849161233887307017428371823608125699892904960841221593344499088996021883972185241854777608212592397013510086894908468466292313

637623513649726535646416995292055104892632668341827716175636313632779328542277

Generación de claves públicas y privadas

```
def generar_clave_ElGamal():
    f = open("grupo.txt","r")
    p = int(f.readline())
    g = int(f.readline())
    q = int(f.readline())
    f.close()

    x = random.randint(1, q - 1)
    f = open("private_key.txt","w")
    f.write(str(x))
    f.close()

    f = open("public_key.txt","w")
    f.write(str(exp_mod(g,x,p)))
    f.close()
```

Cálculo de la firma

```python
def firmar_Schnorr(mensaje: str) -> (int,int):
    f = open("grupo.txt","r")
    p = int(f.readline())
    g = int(f.readline())
    q = int(f.readline())
    f.close()

    f = open("private_key.txt","r")
    x = int(f.readline())
    f.close()

    k = random.randint(1, q - 1)
    r = exp_mod(g,k,p)
    hash = hashlib.sha256()
    hash.update(str(r).encode() + mensaje.encode())
    v = int(hash.hexdigest(),16)
    s = k + v*x
    return (v,s)
```

Clave privada

Verificación de la firma

```
def verificar_firma_Schnorr(mensaje: str, firma: (int,int)) -> bool:
    f = open("grupo.txt","r")
    p = int(f.readline())
    g = int(f.readline())
    f.close()

    f = open("public_key.txt","r")
    y = int(f.readline())
    f.close()

    alpha = exp_mod(g,firma[1],p)
    beta = (alpha * inverso(exp_mod(y,v,p),p)) % p
    hash = hashlib.sha256()
    hash.update(str(beta).encode() + mensaje.encode())
    return v == int(hash.hexdigest(),16)
```

Clave pública

Inverso en módulo p

Utilizando la firma de Schnorr

```python
if __name__ == "__main__":
    generar_clave_ElGamal()
    mensaje = "Alice transfiere 1000 BTC a Bob"
    v,s = firmar_Schnorr(mensaje)
    print("v: ",v)
    print("s: ",s)
    print(verificar_firma_Schnorr(mensaje, (v,s)))
    print(verificar_firma_Schnorr(
        "Alice transfiere 1001 BTC a Bob", (v,s)))
```

Utilizando la firma de Schnorr

v: 19179042201810311532353596372007012380355747
16208713513126393311491981373339

s: 12045255482702459011382783211874212188652965
60544957763944830914817144365700257070609673470
10917105492571023706840659116892843028166673766
56254069195083

Utilizando la firma de Schnorr

private_key.txt

62804259753730070286052654732565555979452984528
83030709195088195059584851103115

Utilizando la firma de Schnorr

public_key.txt

6654015036296758911325351927251947300082815684355273319166245072667201353207430208749628635757237693335634777426951232469200560833582704254094791771768724291546733319366733570192748051813473035250994218826941977930599972433131193990052649463356761308747507542923960179605343088524191915694215541444976688067460799513812811451919812987953800006742662184917373129368515886709450312232703774767415788278749907931964432138540542228411263398696904469922587867878976746887334967282581765959917852786983319559914910949415217913362723265873621256235118165120775825541537880807565471903014128985930088378771090249201666151292

¿Qué ventajas tienen las firmas de Schnorr?

Son más pequeñas que *otras* firmas digitales

Son fáciles de combinar si varios usuarios deben firmar un mensaje

¿Cómo pueden firmar un documento dos usuarios?

Suponga que A y B deben firmar un mensaje m

- Por ejemplo, un pago que debe ser autorizado por ambos usuarios

¿Cómo puede hacer esto usando RSA?

- ¿Es posible tener **una** clave pública para verificar que una firma es válida?

Resolviendo el problema con firmas de Schnorr

Suponemos que:

- La clave privada de A es x_1 y su clave pública es g^{x_1}
- La clave privada de B es x_2 y su clave pública es g^{x_2}

La clave pública para verificar la firma de m por ambos usuarios es $g^{x_1} * g^{x_2} = g^{x_1+x_2}$

Resolviendo el problema con firmas de Schnorr

A y B firman m de la siguiente forma:

1. A genera al azar $k_1 \in \{1, \ldots, q-1\}$ y calcula $r_1 = g^{k_1}$
2. B genera al azar $k_2 \in \{1, \ldots, q-1\}$ y calcula $r_2 = g^{k_2}$
3. *Ambos* calculan $v = h((r_1 * r_2) \| m)$ usando $r_1 * r_2 = g^{k_1+k_2}$ como un string

Resolviendo el problema con firmas de Schnorr

4. A calcula $s_1 = k_1 + v \cdot x_1$ interpretando v como un número natural
5. B calcula $s_2 = k_2 + v \cdot x_2$ interpretando v como un número natural (de la misma forma que A)
6. Ambos calculan $s = s_1 + s_2$, y la firma de m es (v, s)

Resolviendo el problema con firmas de Schnorr

¿Cómo puede un usuario verificar que (v, s) es una firma de m generada por A y B?

¿Cómo se puede generalizar esta idea para n usuarios?