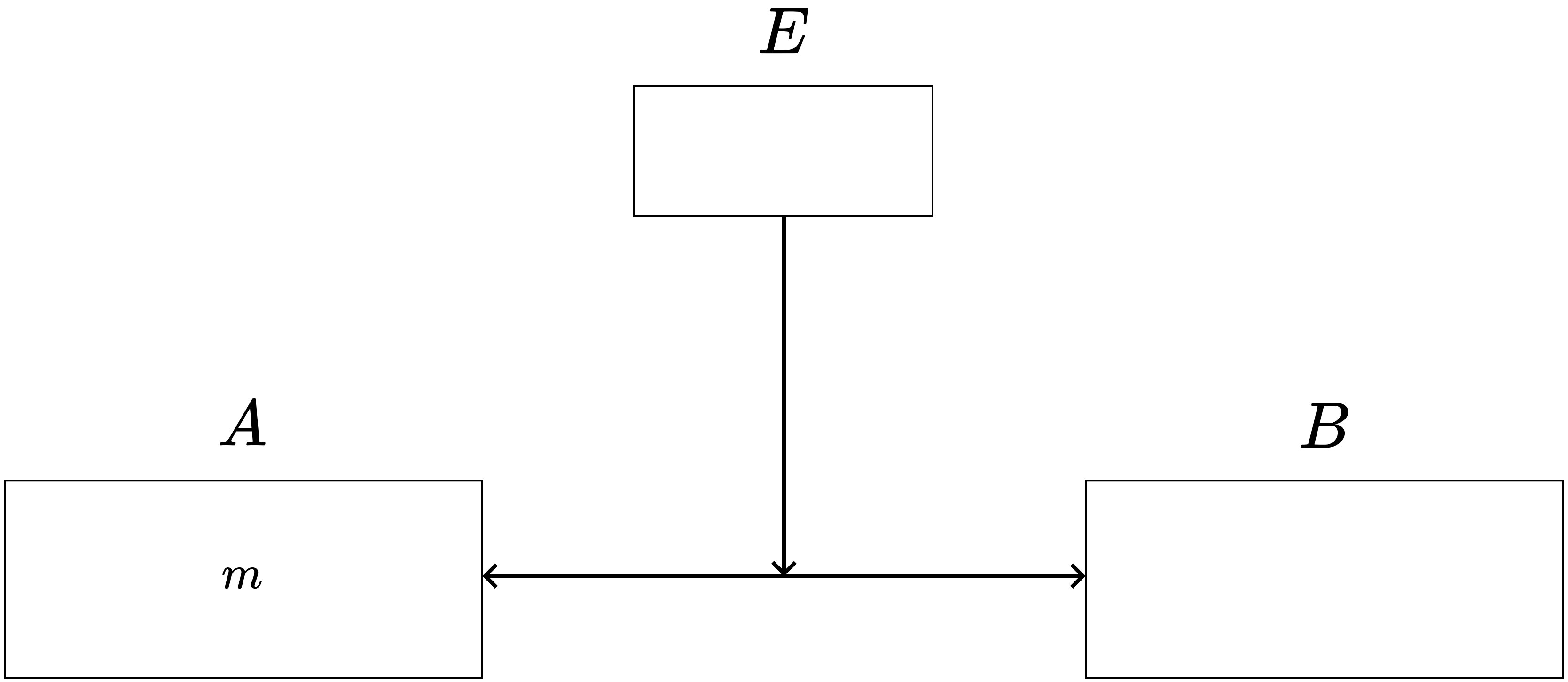
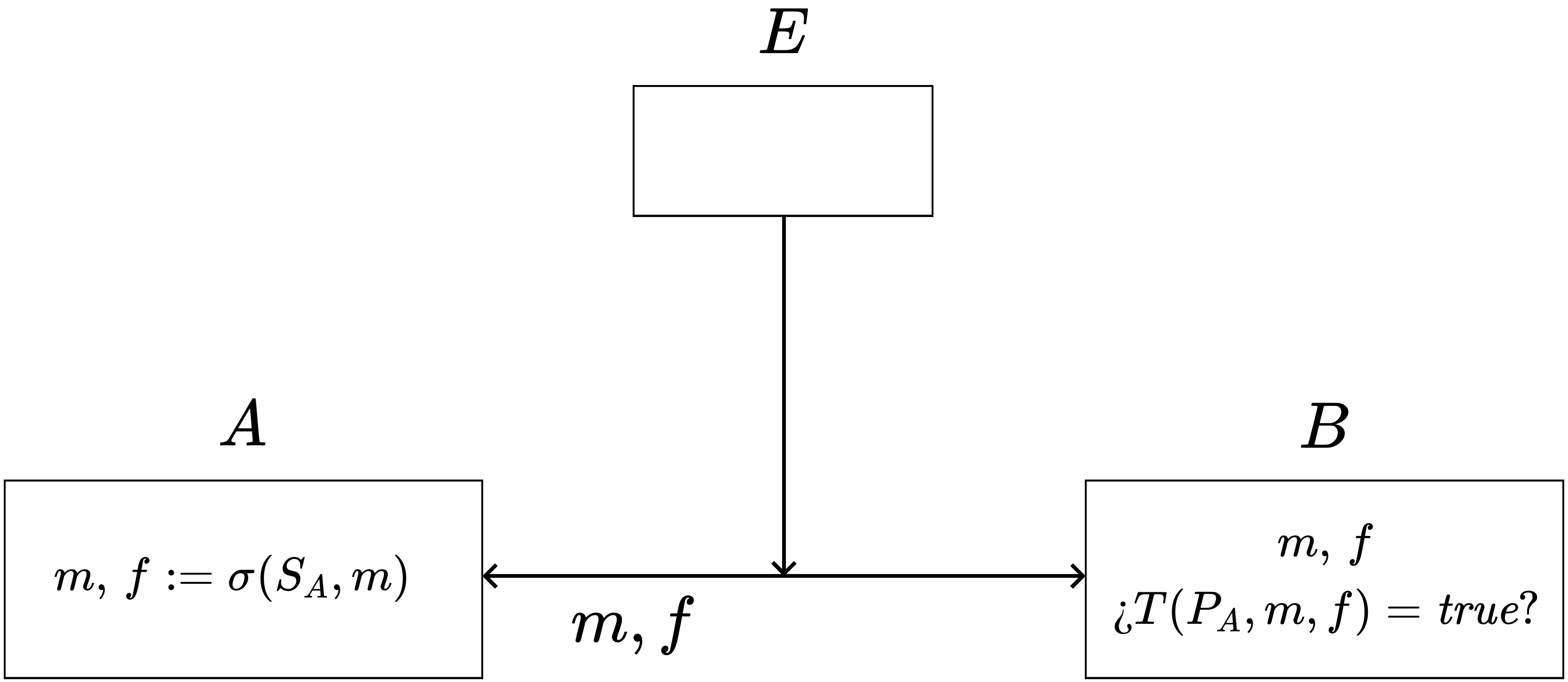
IIC3253

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

$$
\begin{aligned}
Enc_{P_A}(Dec_{S_A}(m)) = \quad & (m^d \bmod N)^e \bmod N \\
&= (m^d)^e \bmod N \\
&= m^{d \cdot e} \bmod N \\
&= (m^e)^d \bmod N \\
&= (m^e \bmod N)^d \bmod N \\
&= Dec_{S_A}(Enc_{P_A}(m)) \\
&= m
\end{aligned}
$$

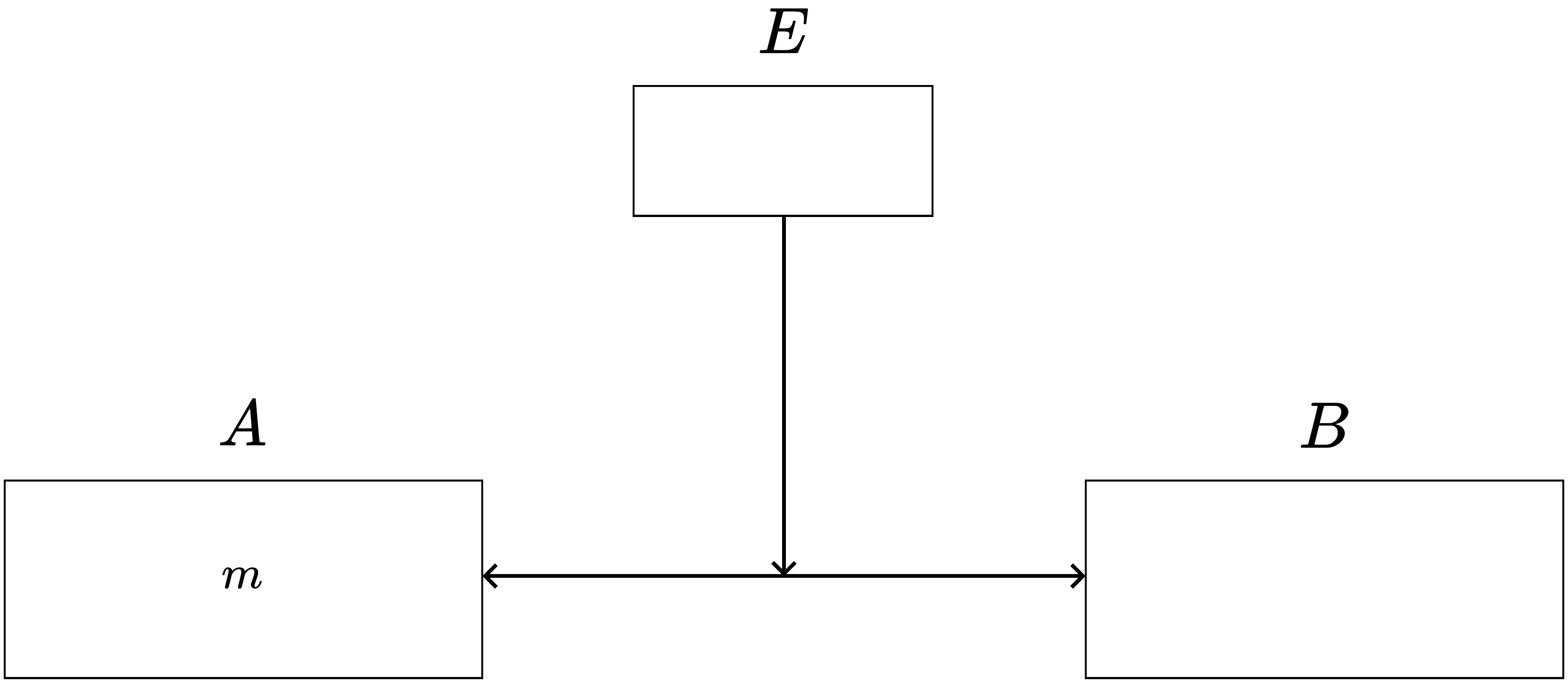
Firmas digitales con RSA

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

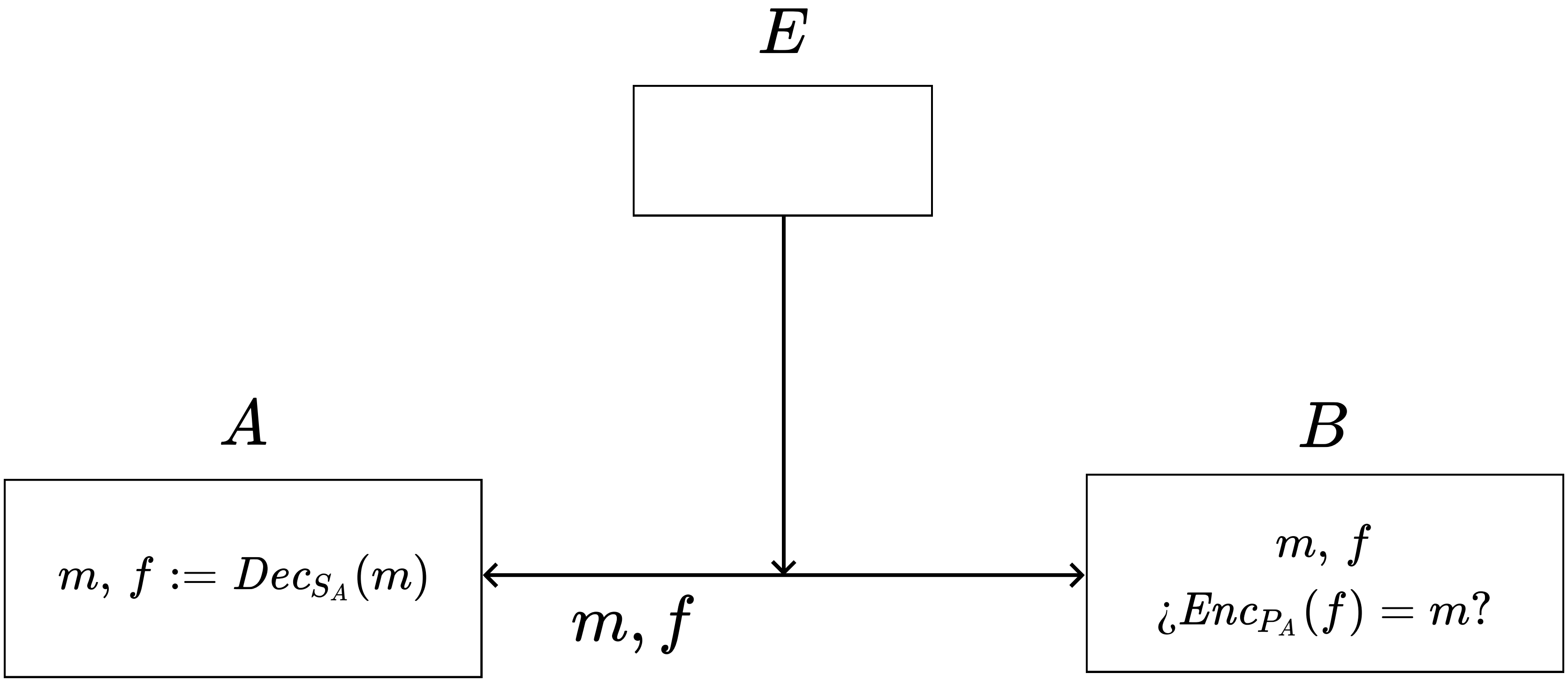
$$f := Dec_{S_A}(m)$$

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

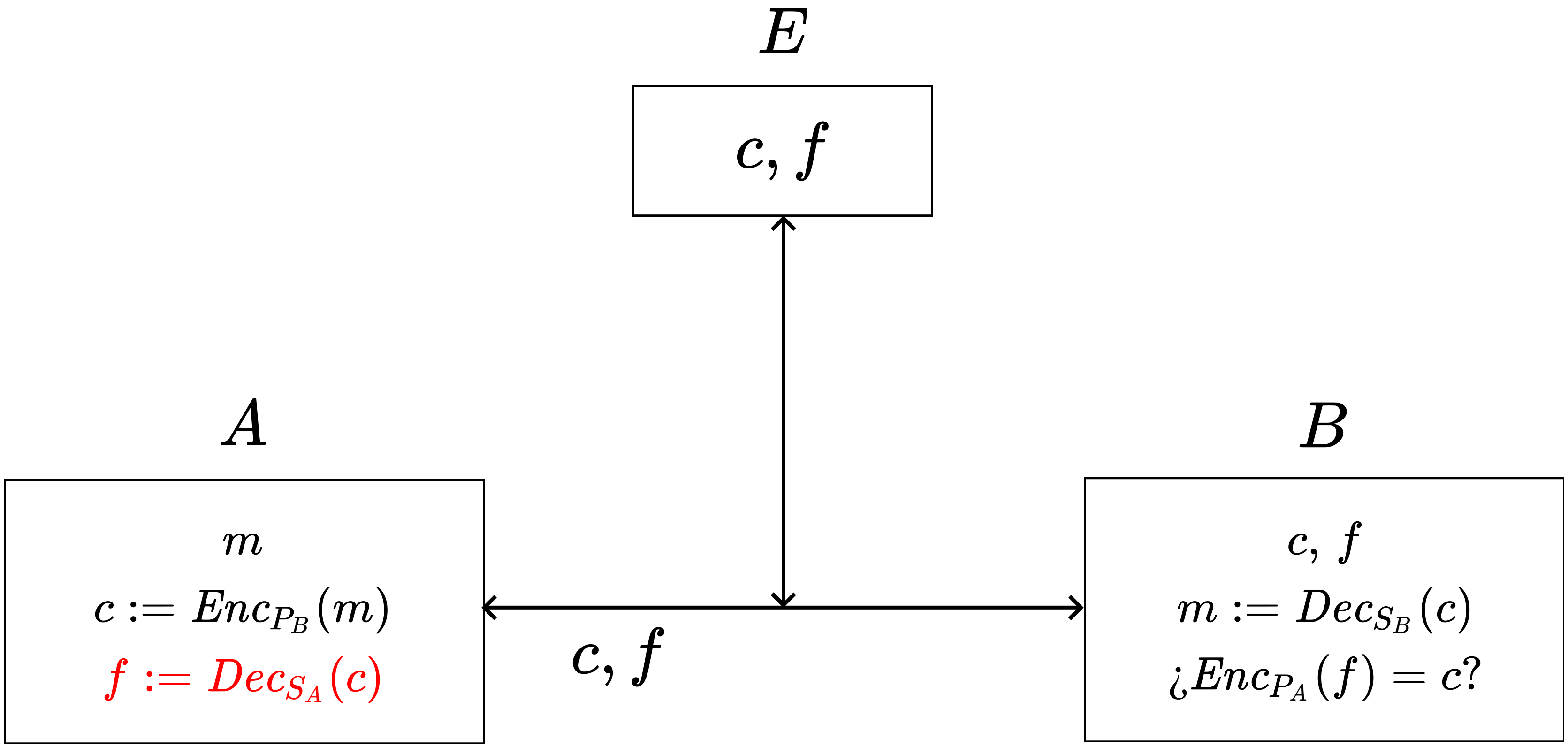
El esquema de firmas digitales con RSA



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 y q un número primo

Además suponemos dada una función de hash h

La definición de las firmas de Schnorr

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

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

La verificación de una firma de Schnorr

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

1. Calcular $\alpha = g^s * y^{q-c}$
2. Verificar si $c = h(\alpha \| m)$

$$g^{r+cx} * y^{q-c} = g^{r+cx} * (g^x)^{q-c} = g^{r+cx+qx-cx} = g^r$$

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 SHA-256 https://www.rfc-editor.org/rfc/rfc5114.html

Network Working Group
Request for Comments: 5114
Category: Informational
M. Lepinski
S. Kent
BBN Technologies
January 2008

Additional Diffie-Hellman Groups for Use with IETF Standards

Status of This Memo

This memo provides information for the Internet community. It does not specify an Internet standard of any kind. Distribution of this memo is unlimited.

Abstract

This document describes eight Diffie-Hellman groups that can be used in conjunction with IETF protocols to provide security for Internet communications. The groups allow implementers to use the same groups with a variety of security protocols, e.g., SMIME, Secure SHell (SSH), Transport Layer Security (TLS), and Internet Key Exchange (IKE).

All of these groups comply in form and structure with relevant standards from ISO, ANSI, NIST, and the IEEE. These groups are compatible with all IETF standards that make use of Diffie-Hellman or Elliptic Curve Diffie-Hellman cryptography.

El archivo grupo.txt

171254583176141379301960419792575778264088323240375085733932929816
426671397476217788024387752387285929683446135893799323484756135034
769321631669738132186983438164632891441853629126025225404949830905
314972329658295365245072698488256583114202993359222957097432675083
225259667739503949192575768420387716327420441424710535098501236058
838158571626669177751934961573726561955583057270098912760065140004
093658772181713883199238963093777917625906143118496429613802248519
404604217104493689272529748703958739363879096722748832953774810081
504758785902705917983505634881680809238046118223875201980540029906
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(pow(g, x, p)))
    f.close()
```

Cálculo de la firma

```python
def firmar_Schnorr(mensaje):
    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()

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

Clave privada

Verificación de la firma

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

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

    alpha = pow(g, firma[1], p)
    beta = (alpha * pow(y, q - firma[0], p)) % p
    hash = hashlib.sha256()
    hash.update(str(beta).encode() + mensaje.encode())
    return firma[0] == int(hash.hexdigest(), 16)
```

Clave pública

Utilizando la firma de Schnorr

```python
if __name__ == "__main__":
    generar_clave_ElGamal()
    mensaje = "Alice transfiere 1000 BTC a Bob"
    c1, s1 = firmar_Schnorr(mensaje)
    print("c: ", c1)
    print("s: ", s1)
    print(verificar_firma_Schnorr(mensaje, (c1, s1)))
    print(verificar_firma_Schnorr(
        "Alice transfiere 1001 BTC a Bob", (c1, s1)))
```

Utilizando la firma de Schnorr

c: 19179042201810311532353596372007012380355747
16208713513126393311491981373339

s: 12045255482702459011382783211874212188652965
6054495776394483091481714436570025707060967347
0109171054925710237068406591168928430281666737
66562540691950833

True

False

Utilizando la firma de Schnorr

private_key.txt

62804259753730070286052654732565559794529845288303070919508819505958485103115

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:

- Firmas de anillo
- Firmas múltiples

Firmas de anillo

Firmas de anillo

Esquema para firmar que permite a un usuario seleccionar un grupo de posibles firmantes (anillo) y generar una firma que:

- Prueba que alguien del grupo firmó
- No revela quién del grupo lo hizo

Firmas de anillo: propiedadas

- **Anonimato del firmante**: imposible identificar al firmante dentro del grupo
- **Verificabilidad**: cualquiera puede verificar que la firma es válida y que fue generada por alguien del grupo
- **Sin coordinación**: los otros miembros del grupo no necesitan dar su consentimiento para construir la firma

Firmas de anillo: aplicaciones

- Revelar un secreto
- Transacciones anónimas en criptomonedas
- Votación electrónica

¿Ve algún problema al tratar de usar las firmas de anillo para votación electrónica?

Firmas de anillo para votación electrónica

- **Linkable ring signatures:** es posible detectar si dos firmas de anillo fueron hechas por el mismo usuario
- **Traceable ring signatures:** si dos firmas de anillo fueron hechas por el mismo usuario, es posible detectar quién lo hizo

Un protocolo para las firmas de anillo

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

- G, g, q y h son públicos

Tenemos un grupo de cuatro integrantes: 1, 2, 3, 4

- Cada integrante i tiene clave secreta x_i y clave pública $y_i = g^{x_i}$

Un protocolo para las firmas de anillo

Suponemos que el usuario 1 quiere generar una firma de anillo de un mensaje m en el grupo $1, 2, 3, 4$

Firma de Schnorr

1

$r_1 \in \{1, \ldots, q-1\}$

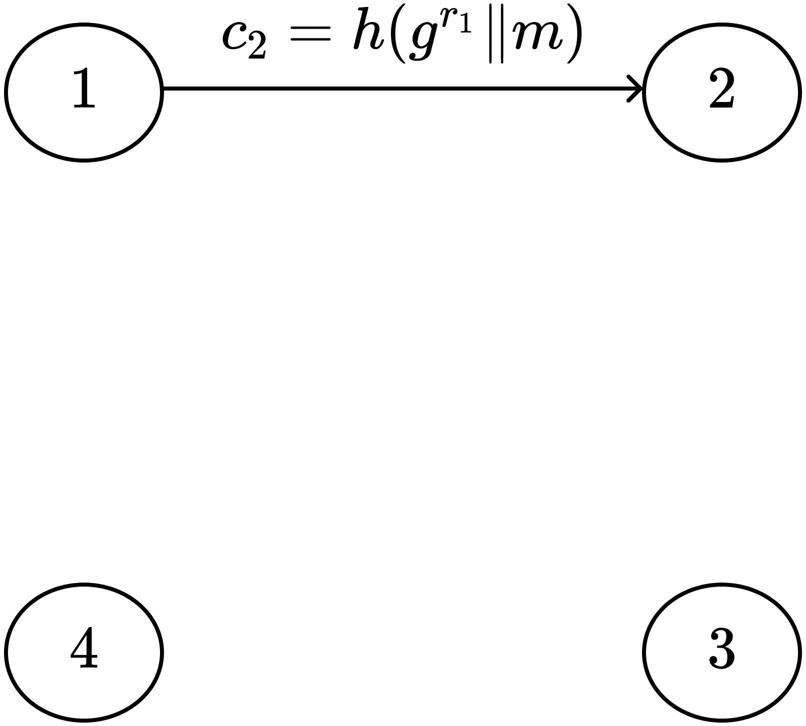
$c_1 = h(g^{r_1} \| m)$

$s_1 = r_1 + c_1 \cdot x_1$

Firma: (c_1, s_1)

Verificación: $c_1 = h(g^{s_1} * y_1^{q-c_1} \| m)$

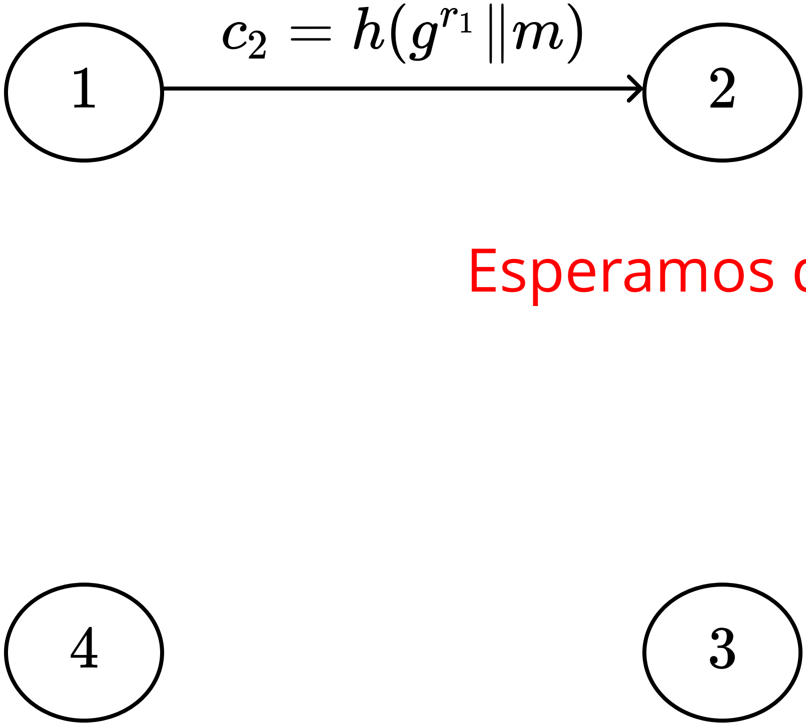
$r_1 \in \{1, \ldots, q-1\}$



$r_1 \in \{1, \ldots, q-1\}$

$s_2 \in \{1, \ldots, q-1\}$

$g^{r_2} = g^{s_2} * y_2^{q-c_2}$

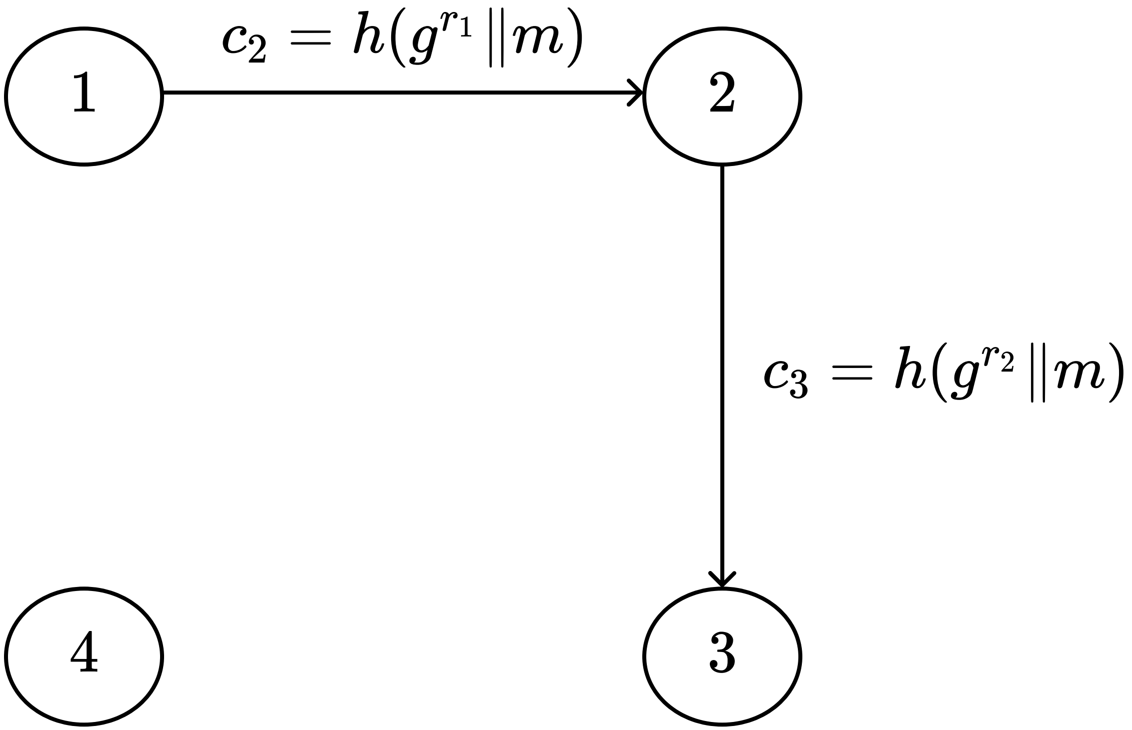


Esperamos que $g^{s_2} = g^{r_2} * y_2^{c_2}$

$r_1 \in \{1, \ldots, q-1\}$

$s_2 \in \{1, \ldots, q-1\}$

$g^{r_2} = g^{s_2} * y_2^{q-c_2}$



$r_1 \in \{1, \ldots, q-1\}$

$s_2 \in \{1, \ldots, q-1\}$

$g^{r_2} = g^{s_2} * y_2^{q-c_2}$

$$
\begin{tikzcd}
1 \arrow[r, "c_2 = h(g^{r_1} \| m)"] & 2 \arrow[d, "c_3 = h(g^{r_2} \| m)"] \\
4 & 3
\end{tikzcd}
$$

$s_3 \in \{1, \ldots, q-1\}$

$g^{r_3} = g^{s_3} * y_3^{q-c_3}$

Esperamos que $g^{s_3} = g^{r_3} * y_3^{c_3}$

$$r_1 \in \{1, \ldots, q-1\}$$

$$s_2 \in \{1, \ldots, q-1\}$$

$$g^{r_2} = g^{s_2} * y_2^{q-c_2}$$

$$\begin{tikzcd} 1 \arrow[r, "c_2 = h(g^{r_1} \| m)"] & 2 \arrow[d, "c_3 = h(g^{r_2} \| m)"] \\ 4 & 3 \arrow[l, "c_4 = h(g^{r_3} \| m)"'] \end{tikzcd}$$

Esperamos que $g^{s_4} = g^{r_4} * y_4^{c_4}$

$$s_4 \in \{1, \ldots, q-1\}$$

$$s_3 \in \{1, \ldots, q-1\}$$

$$g^{r_4} = g^{s_4} * y_4^{q-c_4}$$

$$g^{r_3} = g^{s_3} * y_3^{q-c_3}$$

$r_1 \in \{1, \ldots, q-1\}$

$\boxed{s_1 = r_1 + c_1 \cdot x_1}$

$s_2 \in \{1, \ldots, q-1\}$

$g^{r_2} = g^{s_2} * y_2^{q-c_2}$

$$
\begin{tikzcd}
1 \arrow[r, "c_2 = h(g^{r_1} \| m)"] & 2 \arrow[d, "c_3 = h(g^{r_2} \| m)"] \\
4 \arrow[u, "c_1 = h(g^{r_4} \| m)"] & 3 \arrow[l, "c_4 = h(g^{r_3} \| m)"]
\end{tikzcd}
$$

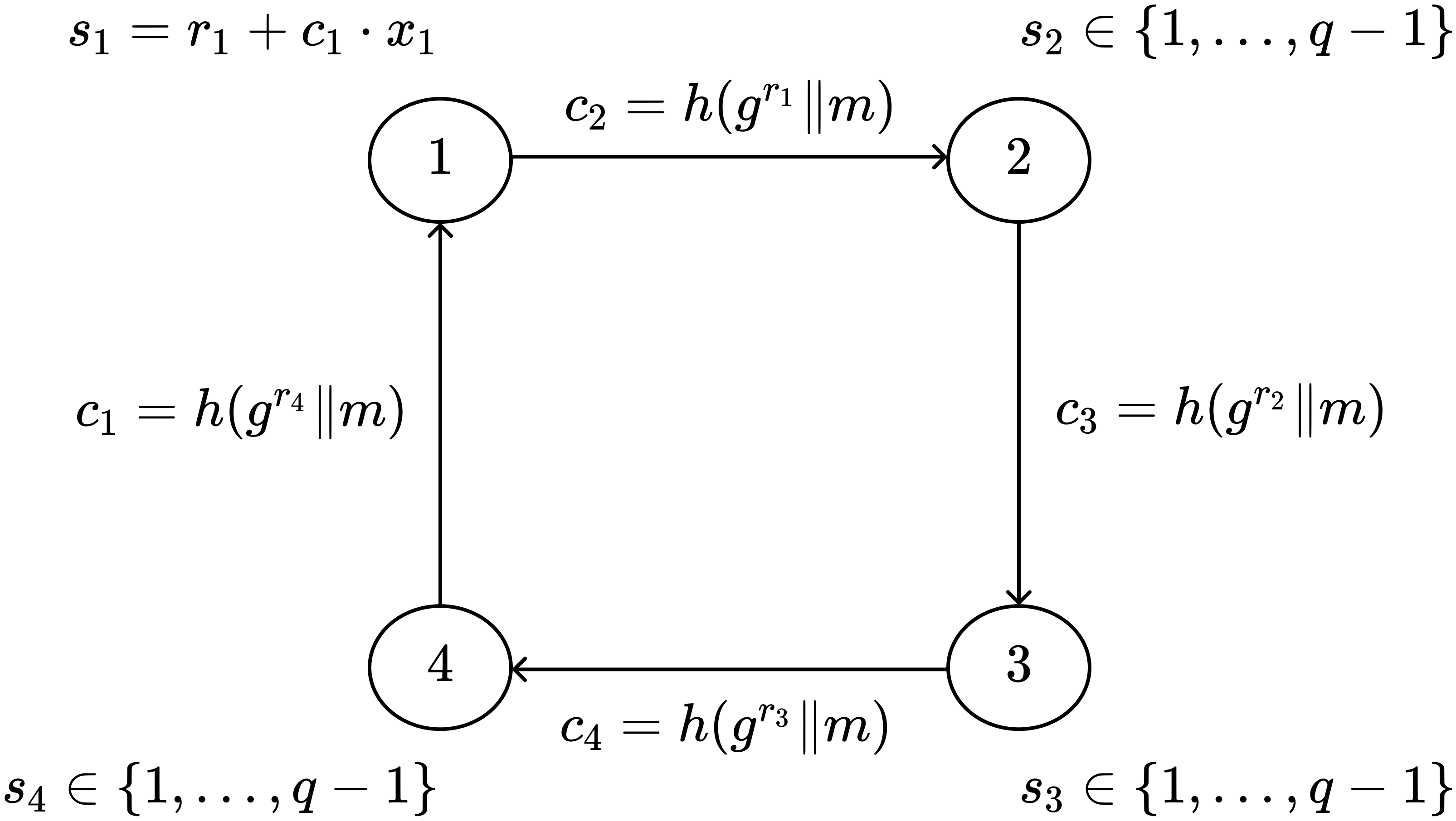
$s_4 \in \{1, \ldots, q-1\}$

$g^{r_4} = g^{s_4} * y_4^{q-c_4}$

$s_3 \in \{1, \ldots, q-1\}$

$g^{r_3} = g^{s_3} * y_3^{q-c_3}$

Firma: $(c_1, s_1, c_2, s_2, c_3, s_3, c_4, s_4)$



Firma: $(c_1, s_1, c_2, s_2, c_3, s_3, c_4, s_4)$

Verificación:

$$c_2 = h(g^{s_1} * y_1^{q-c_1} \| m)$$

$$c_3 = h(g^{s_2} * y_2^{q-c_2} \| m)$$

$$c_4 = h(g^{s_3} * y_2^{q-c_3} \| m)$$

$$c_1 = h(g^{s_4} * y_2^{q-c_4} \| m)$$

¿Es posible descubrir quién firmó el mensaje?

Firmas multiples

¿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_A y su clave pública es g^{x_A}
- La clave privada de B es x_B y su clave pública es g^{x_B}

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

Resolviendo el problema con firmas de Schnorr

A y B firman m de la siguiente forma:

1. A genera al azar $r_A \in \{1, \ldots, q-1\}$ y calcula $R_A = g^{r_A}$
2. B genera al azar $r_B \in \{1, \ldots, q-1\}$ y calcula $R_B = g^{r_B}$
3. *Ambos* calculan $c = h((R_A * R_B) \| m)$ usando $R_A * R_B = g^{r_A + r_B}$ como un string

Resolviendo el problema con firmas de Schnorr

4. A calcula $s_A = r_A + c \cdot x_A$ interpretando c como un número natural

5. B calcula $s_B = r_B + c \cdot x_B$ interpretando c como un número natural (de la misma forma que A)

6. Ambos calculan $s = s_A + s_B$, y la firma de m es (c, s)

Resolviendo el problema con firmas de Schnorr

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

¿Cómo se puede generalizar esta idea para n usuarios?