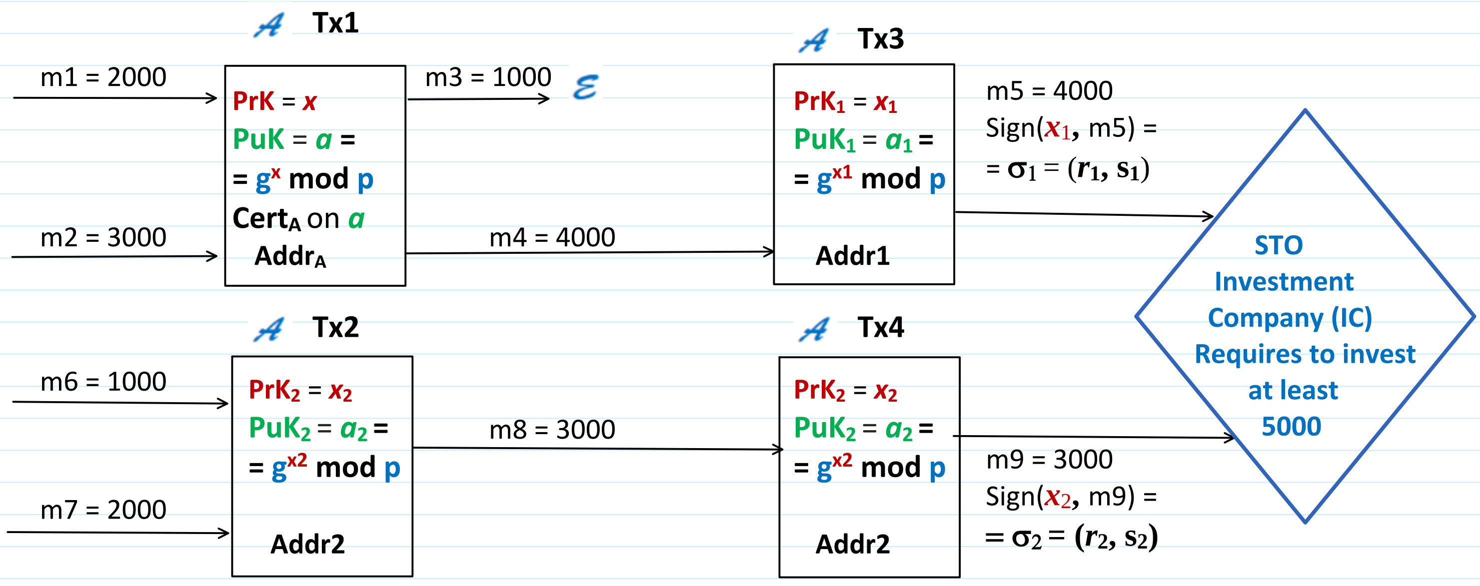


In Monero blockchain for anonymization Alice is using Ring Signature, instead procedure presented above.
It is interresting to compare the realization effectivity of procedure presented above and procedure based on Ring Signature.

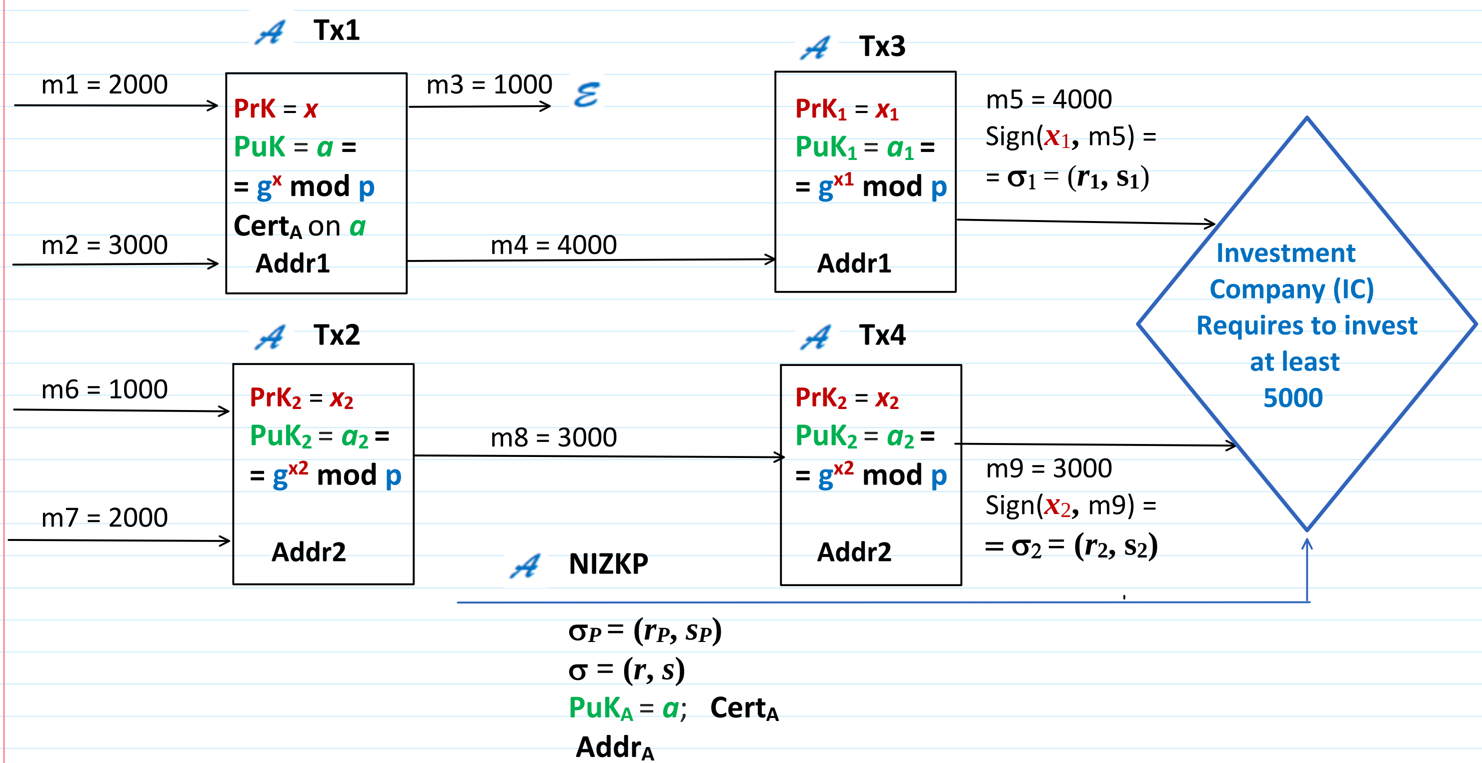
Compare realization effectivity of DEF Schnorr multisignature with ECC ring signature computing the number of
Discrete Exponent Function Operations - DEFO: $a = g^u \bmod p$
Elliptic Curve Cryptography Operations - ECCO: EC point multiplication by integer $z*G = P$.

Anonymity disclosing for Investment Company (IC)
Deanonymization for Authenticity

Alice has private key **$PrK_A = x$** and public key **$PuK_A = a = g^x \bmod p$**.
Alice has a certificate **$Cert_A$** issued by Certificate Authority (CA) on her **$PuK_A = a$**.
Alice must prove to **IC** that $PuK_1 = a_1 = g^{x1} \bmod p$ and $PuK_2 = a_2 = g^{x2} \bmod p$ together with **Addr1** and **Addr2** belongs to her.
This means that she must prove that she knows x_1 and x_2 corresponding to
$PuK_1 = a_1 = g^{x1} \bmod p$ and $PuK_2 = a_2 = g^{x1} \bmod p$.
To save the computation resources **Alice** does not proves the knowledge of every x_1 and x_2.
Alice proves that she knows $x_{12} = x_1 + x_2$ instead since to guess x_{12} without the knowledge of x_1 and x_2 is infeasible.

Alice realizes the Non-Interactve Zero knowledge Proof (NIZKP) of knowledge of x_{12}.
Then she computes h-value: $H' = \mathrm{H}(a_1||\mathrm{Addr1}||a_2||\mathrm{Addr2}||r_p)$
$u \leftarrow \mathrm{randi}(p-1)$.
$r_p = g^u \bmod p$.
$h' = \mathrm{H}(H'||r_p) = \mathrm{H}(a_1||\mathrm{Addr1}||a_2||\mathrm{Addr2}||r_p)$.
$s_p = u + x_{12}h' \bmod (p-1)$.
Alice sends the value $\sigma_P = (r_P, s_P)$ to **IC**.
To reveal her identity **Alice** signs σ_P with her $PrK_A = x$ which corresponds to her $= g^x \bmod p$ and sends her Certificare $\mathbf{Cert_A}$ to **IC**.
$\mathrm{Sign}(x, \sigma_P) = \sigma = (r, s)$.

v <-- randi(p-1).
$r = g^v \bmod p$.
$h = H(\sigma_P || r)$.
$s = v + xh \bmod (p-1)$.
Alice sends the value $\sigma = (r, s)$ to **IC**.



1. **IC** verifies if s =/= S_{12}.
2. **IC** verifies **Cert$_A$** on a and **Alice** signature on $\sigma = (r, s)$ on $\sigma_P = (r_P, s_P)$.

$g^s \bmod p = r a^h \bmod p$.
V1 V2

2. **IC** verifies **Alice** signature on $\sigma_P = (r_P, s_P)$ signed by $x_{12} = x_1 + x_2$.
3. **IC** verifies if (Eq. 2) is valid $g^{S12} \bmod p = R_{12} * (a_1)^{h1} * (a_2)^{h2} \bmod p$.
3. **IC** provides **Alice** with STO according to the sum 7000.

Compare deanonymization with deanonymization used for anonymization with Ring Signatures in Monero.

According to the Birthday Paradox the probability **Prob$_2$** to guess x_{12} when x_{12} is a sum mod **p** of two secret numbers x_1 and x_2 having n=2040 bits is negligible, i.e.

Prob$_2$ $< 2^{-n/2}$. Lina

In the case of sum of k private keys the probability **Prob$_k$** satisfies inequality

Prob$_k$ $< 2^{-n/k}$.

If verification passes then **IC** transfers the interest on investments to **Alice** account.
The material regarding **NIZKP** is included in schemes and explanation is presented below.

Additional material: of Non-Interactive Zero Knowledge Proof (**NIZKP**).

The technique presented above has an essential flaw.
The anyone having **PrK$_{Im}$** = x_{Im} and public key **PuK$_{Im}$** = a_{Im} can impersonate the actual **Addr1** and **Addr2** holder and redirect the interest on investments to his/her account by creating new **Addr$_{Im}$** = **F**(**PuK$_{Im}$**) by obtaining the certificate **Cert$_{Im}$** on **PuK$_{Im}$**.
Then **Impersonator** having Tx2 and Tx4 data together with **PuK$_1$** = a_1 and **PuK$_2$** = a_2 can sign Tx2 and Tx4 with his/her **PrK$_{Im}$** by computing $\sigma_{Im} = (r_{Im}, s_{Im})$.
Then **Impersonator** sends ($\sigma_{Im} = (r_{Im}, s_{Im})$, **PuK$_{Im}$** = a_{Im}, **Cert$_{Im}$** and **Addr$_{Im}$**) as actual **Addr1** and **Addr2** holder **Alice** did.
After **IC** verification the **Impersonator** is waiting when **IC** transfers the interest on investments to his/her account represented by **Addr$_{Im}$**.

The solution of this problem is the realization of Non-Interactive Zero Knowledge Proof (**NIZKP**) by Alice proving that she knows her generated **PrK$_1$** = x_1 and **PrK$_2$** = x_2.
The **NIZKP** is as an additional operation is included in the schemes above.
The details of **NIZKP** realization is left as an exercise.

Anonymity and authenticity simulation

```
>> p=int64(268435019)
p = 268435019
>> g=2;
```

```
>> x=int64(randi(p-1))
x = 257726155
>> a=mod_exp(g,x,p)
a = 32920391
>> AddrA=hd28('32920391')
AddrA = 126423499
```

```
>> x1=int64(randi(p-1))
x1 = 156758073
>> a1=mod_exp(g,x1,p)
a1 = 15617773
>> Addr1=hd28('15617773')
Addr1 = 32691790
```

```
>> x2=int64(randi(p-1))
x2 = 93240757
>> a2=mod_exp(g,x2,p)
a2 = 92735335
>> Addr2=hd28('92735335')
Addr2 = 186632019
```

```
Tx2='In21=4000||Ex21=4000||Addr1'

>> u1=int64(randi(p-1))
u1 = 50037375
>> r1=mod_exp(g,u1,p)
r1 = 32904517
>> con=concat(Tx2,r1)
con = In21=4000||Ex21=4000||Addr132904517
>> h1=hd28(con)
h1 = 64943318
>> s1=mod(u1+x1*h1,p-1)
```

```
Tx4='In41=3000||Ex41=3000||Addr1'

>> u2=int64(randi(p-1))
u2 = 190308111
>> r2=mod_exp(g,u2,p)
r2 = 22463608
>> con=concat(Tx4,r2)
con = In41=3000||Ex41=3000||Addr122463608
>> h2=hd28(con)
h2 = 26322703
>> s2=mod(u2+x2*h2,p-1)
```

```
>> h1=hd28(con)
h1 = 64943318
>> s1=mod(u1+x1*h1,p-1)
s1 = 234649183
```

```
>> h2=hd28(con)
h2 = 26322703
>> s2=mod(u2+x2*h2,p-1)
s2 = 61742096
```

```
>> R12=mod(r1*r2,p)
R12 = 92919544
>> S12=mod(s1+s2,p-1)
S12 = 27956261
>> g_S12=mod_exp(g,S12,p)
g_S12 = 91640974 = V1
```

```
>> a1_h1=mod_exp(a1,h1,p)
a1_h1 = 168145239
>> a2_h2=mod_exp(a2,h2,p)
a2_h2 = 55254133
```

```
>> R12ma1_h1=mod(R12*a1_h1,p)
R12ma1_h1 = 241090947
>> R12ma1_h1ma2_h2=mod(R12ma1_h1*a2_h2,p)
R12ma1_h1ma2_h2 = 91640974 = 91640974 = V2
```

Schnorr-Multi-Signature is valid since V1 = V2 = **91640974**

Deanonymization against **IC**

```
> con1=concat(a1,Addr1)
con1 = 1561777332691790
>> con2=concat(a2,Addr2)
con2 = 92735335186632019
>> con12=concat(con1,con2)
con12 = 156177733269179092735335186632019
>> HH=hd28(con12)
HH = 150477396
```

```
>> u=int64(randi(p-1))
u = 218160208
>> r=mod_exp(g,u,p)
r = 76047239
>> conHHr=concat(HH,r)              %   H'||r
conHHr = 15047739676047239         % HH==H'
>> h=hd28(conHHr)
h = 114895503
>> s=mod(u+x*h,p-1)
s = 107897009
```

$g^s \bmod p = r a^h \bmod p$. (Eq.1)

V1 V2

```
> g_s=mod_exp(g,s,p)
g_s = 18634187
>> V1=g_s
V1 = 18634187
```

```
>> a_h=mod_exp(a,h,p)
a_h = 202702734
>> V2=mod(r*a_h,p)
V2 = 18634187
```

Till this place