ElGamal homomorphic encryption: $PP = (p, g)$

B: $PrK_B = y$; $PuK_B = b$.

$b = g^y \bmod p$

A: $PrK_A = x$; $PuK = a$.

$a = g^x \bmod p$

$PuK_A = a$. $n$ – message $r \leftarrow$ randi

$Enc(a, r, n) = c = (E, D)$ ⟶ A: $Dec(x, c) = n$

$E = n \cdot a^r \bmod p$; $D = g^r \bmod p$

1) $D^{-x} \bmod p$

2) $E \cdot D^{-x} \bmod p = n$

$$Enc: \mathcal{L}_{p-1}^{r} \times \mathcal{L}_p^{n} \longleftrightarrow \mathcal{L}_p^{*E} \times \mathcal{L}_p^{*D}$$

Let $n_1, n_2$ – are messages to be encrypted: $n_1 \cdot n_2 \bmod p = n_{12}$

$r_1, r_2$ – are random numbers for encryption

$Enc(a, r_1, n_1) = c_1 = (E_1, D_1)$

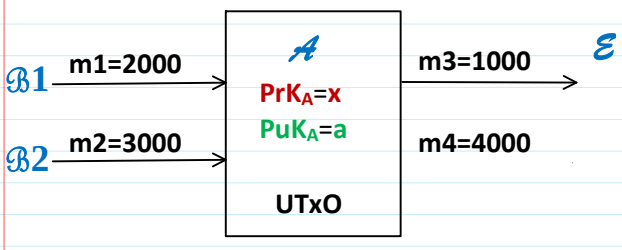
$Enc(a, r_2, n_2) = c_2 = (E_2, D_2)$

$Enc(a, (r_1 + r_2) \bmod (p-1), n_1 \cdot n_2 \bmod p) =$

$= c_1 * c_2 = c_{12} = (E_1 * E_2 \bmod p, D_1 * D_2 \bmod p) = (E_{12}, D_{12})$

$Dec(x, c_{12}) = n_1 \cdot n_2 \bmod p = n_{12}$.

Transaction - **Tx** in blockchain




Balance: $m_1 + m_2 = m_3 + m_4$

How to verify balance when inputs and expenses are encrypted?

Assume that $n_1 = g^{m_1} \bmod p$; $n_2 = g^{m_2} \bmod p$ ⇒ $n_1 * n_2 = g^{m_1 + m_2} \bmod p$

$n_3 = g^{m_3} \bmod p$; $n_4 = g^{m_4} \bmod p$ ⇒ $n_3 * n_4 = g^{m_3 + m_4} \bmod p$

Since $DEF: \mathcal{L}_{p-1} \longleftrightarrow \mathcal{L}_p^*$ is 1-to-1 (bijective) then

if $(m_1 + m_2) \bmod (p-1) = (m_3 + m_4) \bmod (p-1)$

then $n_1 \cdot n_2 \bmod p = n_3 \cdot n_4 \bmod p$

then $n_1 \cdot n_2 \bmod p = n_3 \cdot n_4 \bmod p$

If $m_1 + m_2 < p-1$ & $m_3 + m_4 < p-1$

And $m_1 + m_2 = m_3 + m_4 \Leftrightarrow n_1 \cdot n_2 \bmod p = n_3 \cdot n_4 \bmod p$

$p = 11;\ g = 2:$ $\quad m_1 = 1;\ g^{m_1} \bmod p \rightarrow c_1 \rightarrow Dec(x, g^{m_1}) = g^{m_1} = n_1$; $\quad m_2 = 2;\ g^{m_2} \bmod p \rightarrow c_2 \rightarrow Dec(x, g^{m_2}) = g^{m_2} = n_2$

$(m_1 + m_2) \bmod (p-1) = (1+2) \bmod (p-1) = 3$
mod 10 mod 10

$m_1 = 4,\ m_2 = 9 \Rightarrow (m_1 + m_2) \bmod 10 = (4+9) \bmod 10 = 13 \bmod 10 = 3$

Prevention: $m_1 + m_2 < (p-1)/2$

| $\mathbb{Z}_{p-1}$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | ±5 | -4 | -3 | -2 | -1 |

Let transaction data must be declared to Audit Organization (AA) having key pait=r
**$PrK_{AO}$=z**; **$PuK_{AO}$=e**.

```
>> p=int64(268435019)
p = 268435019
>> g=2;
>>
>> z=int64(randi(p-1))
x = 141076898
>>  a=mod_exp(g,x,p)
e = 116336486
```

```
>> m1=2000;
>> m2=3000;
>> m12=mod(m1+m2,p-1)
m12 = 5000
>> n1=mod_exp(g,m1,p)
n1 = 28125784
>> n2=mod_exp(g,m2,p)
n2 = 222979214
```

% m1+m2=m12=5000 << (p-1)/2

$m_1$: $i_1 \leftarrow randi(\mathbb{Z}_{p-1})$

E1 = $\mathcal{E}_1 = n_1 * e^{i_1} \bmod p$

D1 = $\delta_1 = g^{i_1} \bmod p$

```
>> i1=int64(randi(p-1))
i1 = 256575903
>> e_i1=mod_exp(e,i1,p)
a_i1 = 177744290
>> E1=mod(n1*e_i1,p)
E1 = 78907012
>> D1=mod_exp(g,i1,p)
D1 = 71219017
```

Computations in Cloud data base

```
>> E12=mod(E1*E2,p)
E12 = 248852506
>> D12=mod(D1*D2,p)
D12 = 220753507
```

c12=(E12, D12)

$m_2$: $i_2 \leftarrow randi(\mathbb{Z}_{p-1})$

E2 = $\mathcal{E}_2 = n_2 * e^{i_2} \bmod p$

D2 = ~ $e^{i_2}$ mod p

```
>> i2=int64(randi(p-1))
i2 = 148753825
>> a_i2=mod_exp(e,i2,p)
e_i2 = 206019372
```

E2 = $\varepsilon_2 = n_2 * e^{i_2} \bmod p$

D2 = $\delta_2 = g^{i_2} \bmod p$

```
>> a_i2=mod_exp(e,i2,p)
e_i2 = 206019372
>> E2=mod(n2*e_i2,p)
E2 = 144070332
>> D2=mod_exp(g,i2,p)
D2 = 20873398
```

c12=(E12, D12)

**Audit Organization** decrypts c12 =(E12, D12)

```
>> mz=mod(-z,p-1)        % mines z mod p-1 computation
mz = 127358120
>> mod(mz+z,p-1)         % verification that mz+z mod(p-1)=0
ans = 0
>> D12_mz=mod_exp(D12,mz,p)   % D12^-z computation for decryption
D12_mz = 21824811
>> nnn12=mod(E12*D12_mz,p)    % decryption formula E12* D12^-z mod p = n12 nnn12
nnn12 = 143845522
```

Verification

```
>> nn12=mod(n1*n2,p)
nn12 = 143845522
>> n12=mod_exp(g,m12,p)
n12 = 143845522
```

$n_1 * n_2 = g^{m_1 + m_2} \bmod p$

dlog.m

```
% Finds discrete logarithm value corresponding to exponent value i
% by total scan of i from start by step until fin
% p - is a strong prime (Public Parameter)
% g - is a generator (Public Parameter)
% def - is a discrete exponent function value computed by mod_exp(g,i,p)
%       where dl=i is a searchable value of exponent
%
function dl = dlog(p, g, def, start, step, fin)
 dl=0;
 i=start;
 while i<fin
    ee=mod_exp(g,i,p);
    if ee==def
      dl=i;
      return;
    endif
    i+=step;
 endwhile
 disp('Exponent is not found!');
end
```

```
>> def=nnn12
def = 143845522
>> start=0
start = 0
>> step=100
step = 100
>> fin=9900
fin = 9900
>>
>> dl = dlog(p, g, def, start, step, fin)
dl = 5000
```