Exam content.

1.**Oral part**. You can prepare formulas in advance without comments.

  1.1.Coin flipping.

  1.2.Bit commitment using RSA.

2.**Computation part**. You should provide a computations and write results in the Google drive.

 The training of this part will be realized in 10-th of December during our class.

  2.1.Proxy signature realization.

https://docs.google.com/spreadsheets/d/1PN47UoRWqQtWRAuMf9inR9uRXABi98Ib/edit?usp=sharing&ouid=111502255533491874828&rtpof=true&sd=true

  2.2.Additively-multiplicative encryption realization.

https://docs.google.com/spreadsheets/d/12kEtqRh10RKuUaFVZMIsm2HjUhfKXwKf/edit?usp=sharing&ouid=111502255533491874828&rtpof=true&sd=true

Poster Report (**PR**) presentation will be held in 17-th of December during our class.

PR requirements are placed in:

https://docs.google.com/document/d/1raqTudLCNlLm3wLFCDp_V7QnOg_EFH6d/edit?usp=sharing&ouid=111502255533491874828&rtpof=true&sd=true

PR topic are placed in:

https://docs.google.com/document/d/1KjXlhHhRQJnKnbCbK8crbOxoxy-EaSBf/edit?usp=sharing&ouid=111502255533491874828&rtpof=true&sd=true

Public parameters: **PP** = (***p***, ***g***); p=int64(268435019); g=2;

Proxy signature

Public Parameters $PP = (p, g)$

Key generation and distribution

A: Original Signer.

$PrK_A = x$; $PuK_A = a = g^x \bmod p$

$t \leftarrow randi(p-1)$

$b = g^t \bmod p$

$y = x + t \cdot b \bmod (p-1)$

a, b, y

secure channel

B: Proxy Signer

$PuK_A = a.$

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

Users

(a, b)

$Ver(g^y \stackrel{?}{=} a \cdot b^b \bmod p)$

$g^y = g^{x + t \cdot b} =$

$= g^x \cdot g^{t \cdot b} = a \cdot b^b \bmod p$

Soft – a doc. to be signed

$H(Soft) = h$; $|h| = 256\,b.$

$\xi \leftarrow randi(p-1)$

$$r = g^{\xi} \bmod p$$
$$s = \xi + y\cdot h \bmod (p-1)$$
$$\sigma = (r, s) \quad a, b, \sigma, \text{Soft}$$

1. $Ver(a, b) \stackrel{?}{=} T$
2. $H(\text{Soft}) = h$
3. $Ver(\sigma, h, a, b) \stackrel{?}{=} T$

Verification identity:

$$g^{s} = r\cdot(a\cdot b^{b})^{h} \bmod p$$

$$g^{s} = g^{\xi + y\cdot h} = g^{\xi}\cdot g^{y\cdot h} = r\cdot(g^{y})^{h} = r\cdot(g^{x + t\cdot b})^{h} =$$
$$= r\cdot(g^{x\cdot h + t\cdot b\cdot h}) = r\cdot(g^{x})^{h}\cdot(g^{t})^{b\cdot h} = r\cdot a^{h}\cdot(b^{b})^{h} =$$
$$= r\cdot(a\cdot b^{b})^{h} \bmod p.$$

```
p = int64(268435019);
>> g=2;
> x=int64(randi(p-1))
x = 128831375
>> a=mod_exp(g,x,p)
a = 99834208
```

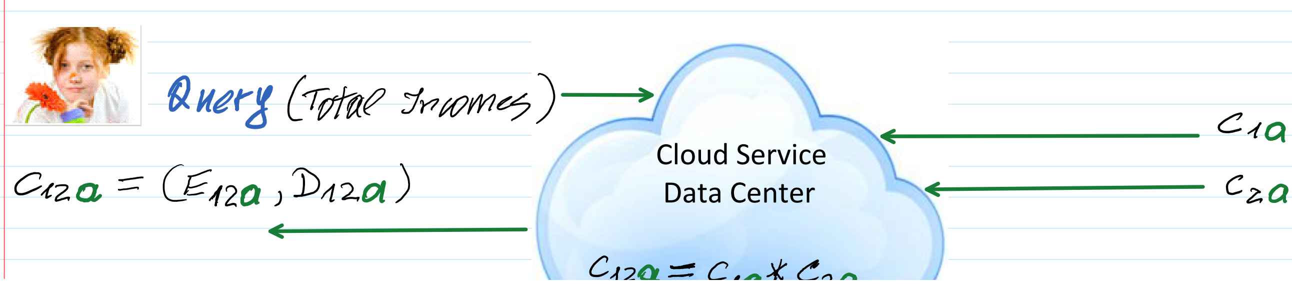
```
>> t=int64(randi(p-1))
t = 54296150
>> b=mod_exp(g,t,p)
b = 267224695
>> y=mod(x+t*b,p-1)
y = 94866523
```

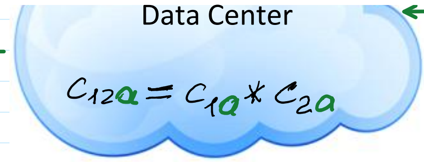
```
>> g_y=mod_exp(g,y,p)
g_y = 81395743
>> V1=g_y
V1 = 81395743
>> b_b=mod_exp(b,b,p)
b_b = 17947996
>> V2=mod(a*b_b,p)
V2 = 81395743
```

```
>> ksy=int64(randi(p-1))
ksy = 5716357
>> r=mod_exp(g,ksy,p)
r = 118257748
>> s=mod(ksy+y*h,p-1)
s = 521536
```

```
>> g_s=mod_exp(g,s,p)
g_s = 115985652
>> V1=g_s
V1 = 115985652
>> b_b=mod_exp(b,b,p)
b_b = 17947996
>> ab_b=mod(a*b_b,p)
ab_b = 81395743
>> ab_b_h=mod_exp(ab_b,h,p)
ab_b_h = 112511772
>> rab_b_h=mod(r*ab_b_h,p)
rab_b_h = 115985652
>> v2=rab_b_h
v2 = 115985652
```

Additively-multiplicative encryption



$c_{12a} = (E_{12a}, D_{12a})$ ← 

← c_{2a}

$$\left.\begin{array}{l} c_{1a} = (E_{1a}, D_{1a}) = (n_1 * a^{i_1}, g^{i_1}) \\ c_{2a} = (E_{2a}, D_{2a}) = (n_2 * a^{i_2}, g^{i_2}) \end{array}\right\} c_{12} = (n_1 * n_2 * a^{i_1+i_2}, g^{i_1+i_2}) = (n_{12} * a^{i}, g^{i})$$

$$i = i_1 + i_2 \mod (p-1)$$

$Dec(x, c_{12a}) = n_{12}$

1. $(D_{12a})^{-x} = (g^{i})^{-x} = g^{-xi} = (g^{x})^{-i} = a^{-i} \mod p$

2. $E_{12a} * (D_{12a})^{-x} \mod p = n_{12} * a^{i} * a^{-i} = n_{12} \mod p$

$$n_{12} = g^{m_1} * g^{m_2} \mod p = g^{m_1+m_2} \mod p.$$

DEF: is one-way function

1) By having p, g and x it is easy to compute $a = g^{x} \mod p$

2) It is infeasible to find x when p, g and a are given!

Zether: Towards Privacy in a Smart Contract World
Financial Cryptography and Data ..., 2020 - Springer

```
>> int64(2^32)
ans = 4 294 967 296
```

The sums $m_1, m_2, ..., m_N$ are restricted in such a way that $m_1 + m_2 + ... + m_N \mod (p-1) < 2^{32}$

To find the sum $m_1 + m_2 = 2000 + 3000 = 5000 \mod (p-1)$

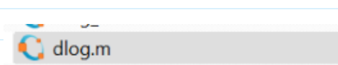
```
>> i1=int64(randi(p-1))
i1 = 39342528
>> a_i1=mod_exp(a,i1,p)
a_i1 = 65358247
>> E1=mod(n1*a_i1,p)
E1 = 69273021
>> D1=mod_exp(g,i1,p)
D1 = 21950001
```

```
>> i2=int64(randi(p-1))
i2 = 67381702
>> a_i2=mod_exp(a,i2,p)
a_i2 = 2020363
>> E2=mod(n2*a_i2,p)
E2 = 30578084
>> D2=mod_exp(g,i2,p)
D2 = 176258356
```

```
>> E12=mod(E1*E2,p)
E12 = 219346681
>> D12=mod(D1*D2,p)
D12 = 108032227
```

```
>> mx=mod(-x,p-1)
mx = 139603643
>> xmx=mod(x+mx,p-1)
xmx = 0
>>
>> D12_mx=mod_exp(D12,mx,p)
D12_mx = 227392947
>> nn12=mod(E12*D12_mx,p)
nn12 = 143845522
>> nnn12=mod(n1*n2,p)
nnn12 = 143845522
```

% Finds discrete logarithm value corresponding to exponent value ***i***
% by total scan of ***i*** from **start** by **step** until **fin**



```
%                                    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)
% 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
```

```
> start=0;
>> step=1;
>> fin=9900;
>> def=nn12
def = 143845522
>>  dl = dlog(p, g, def, start, step, fin)
dl = 5000
```

Till this place

Subliminal Channel - Steganography
Using Schnorr Signature

M - mashing message to be signed.

$n' = k$ - secret message to be sent.

gcd(**z**, **p**-1) = 1, then $\mathbf{z}^{-1}$ mod(**p**-1) exists.

$$\left.\begin{aligned} r &= g^k \bmod p \\ h &= H(M\|r) \\ s &= h + z * k \bmod (p-1) \end{aligned}\right\} \xrightarrow{M,\ \sigma=(r,s)} B: h = H(M\|r)$$

$$\begin{cases} V1 = g^s \bmod p \\ V2 = r * c^h \bmod p \end{cases} \rightarrow V1 \overset{?}{=} V2$$

$$s - h = z * k \quad / * z^{-1}$$

$$k = (s-h) * z^{-1} \bmod (p-1) = n'$$