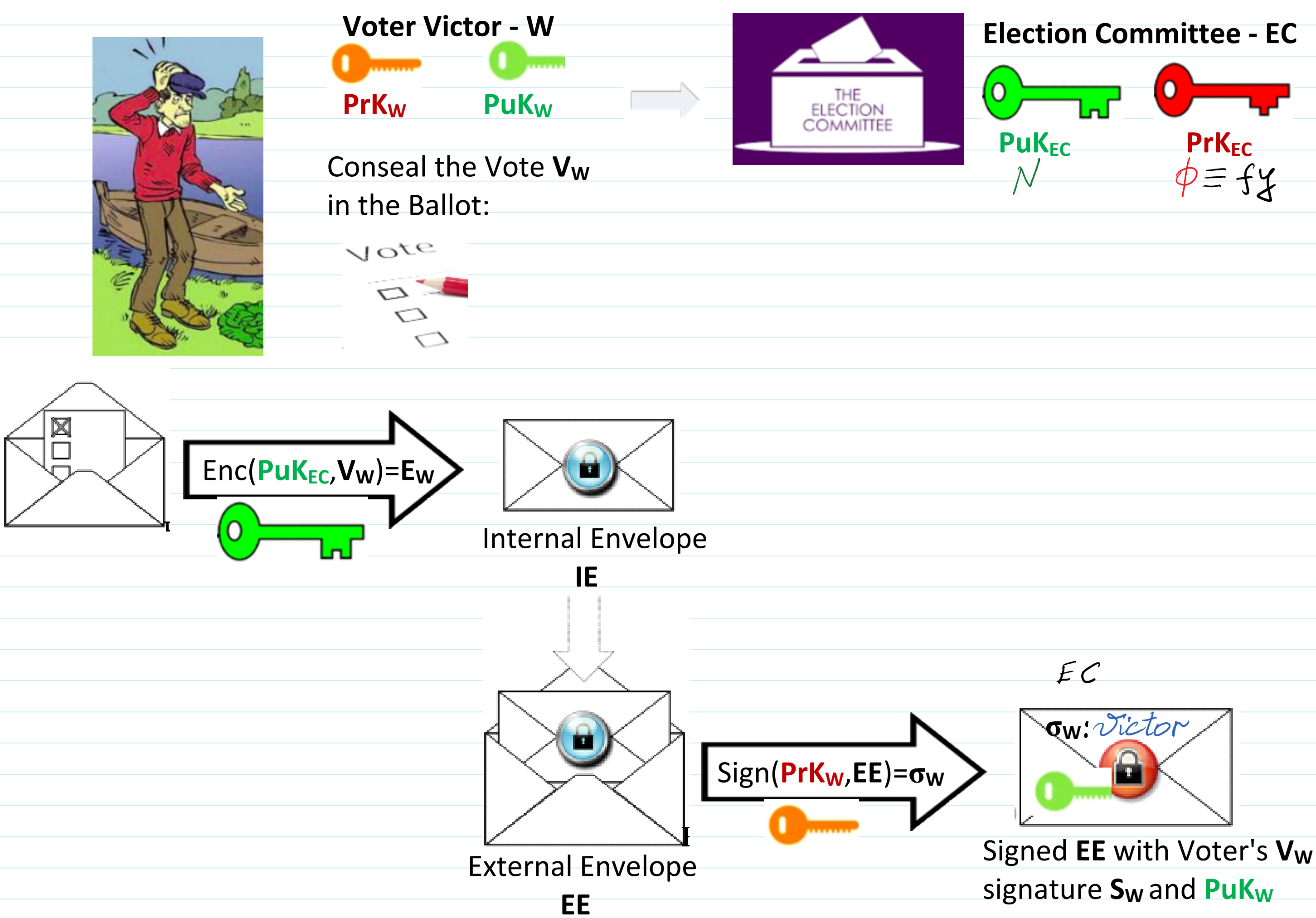
Koliokviumas vyks Balandžio 6d., 17:30, per Zoom (kontaktinis 142 kab., Studentų 50).
Jums reikės realizuoti eBalsavimo sistemą.
Dalis balsų bus pateikta paštu, nors šis balsavimo būdas buvo kritikuojamas.
Jums reikės užpildyti balsavimo lentelę Google drive:
https://docs.google.com/spreadsheets/d/14SygoaSn-QLzafEetSbwrMT4-Qv05JlI/edit?usp=sharing&ouid=111502255533491874828&rtpof=true&sd=true

Lentelėje pataisykite savo Pavardė Vardas į Pa. Vardas.
Pirmos 2 eilutės yra kaip pvz.
Mokymasį kaip tai padaryti pradėsime šiandien.

eVoting System must guarantee:
- Conseal the Vote
- Conseal the Ballots



After eVoting it is the time to calculate the results.
1. **EC** verifies Voters V_W PuK_W and if PuK_W is registered in **EC** database then goes to step 2.
2. **EC** verifies PuK_W certificate and if it is valid then goes to step 3.

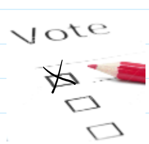
3. **EC** verifies signature **σ_W** on **EE** and if it is valid then extracts **IE** and prceeds with ballots computation.

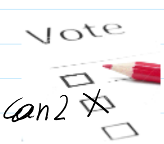
```
>> p=109;
>> q=127;
>> N=p*q                                 % PuK=N=13843
N = 13843
                                         % |N|=14 bits
>> N_2=int64(N*N)
N_2 = 191628649
>> dec2bin(N_2)
ans = 1011 0110 1100 0000 0101 0110 1001

>> fy=(p-1)*(q-1)
fy = 13608                               % PrK=fy
```

Ballots computation

1. Collects all encrypted votes: $(E_W, E_2, E_3, \ldots, E_M)$.
   Number of Voters is M.
2. Multiplies all encrypted votes
   $$E = E_W \cdot E_2 \cdot E_3 \cdot \ldots \cdot E_M$$
3. Decrypts E
   $$Dec(PrK_{EC}, E) = V.$$

If there are 2 candidates: Can1 := 0 ; Can2 := 1

$0 \equiv$ Can1 

$1 \equiv$ Can2 

When using Paillier homomorphic encryption

$$Dec(PrK_{EC}, E) = V = V_W + V_2 + V_3 + \ldots + V_M$$

Let V_{k1} is a number of votes dedicated to Can 1.
Let V_{k2} is a number of votes dedicated to Can 2: $\Rightarrow V = V_{k2}$.
Then the number of votes for Can1: $M - V$

We used homomorphic encryption property:

$$Enc(PuK_{EC}, V_W + V_2 + V_3 + \ldots + V_M) = E_W \cdot E_2 \cdot E_3 \cdot \ldots \cdot E_M = E$$

where $E_W = Enc(PuK_{EC}, V_W)$, $E_2 = Enc(PuK_{EC}, V_2)$, ...

$$Dec(PrK_{EC}, E) = Dec(PrK_{EC}, E_W \cdot E_2 \cdot E_3 \cdot \ldots \cdot E_M) = V_W + V_2 + V_3 + \ldots + V_M = V$$

Let: **K** - be a number of Candidates (Can);
M - be a number of Voters (V);

For every candidate **Can1**, **Can2**, ..., **CanK** the **Vote** is encoded by certain integer number is assigned.
Since all **Votes** are encrypted by every **Voter** using Paillier homomorphic encryption scheme, therefore the maximal sum of **Votes** must not increase **PuK** value **N**.
It is due to the property of Paillier encryption stating that encrypted message $\mathbf{m} \in \mathbf{Z_N} = \{1, 2, 3, \ldots, N-1\}$.
Then due to homomorphic property of Paillier encryption when all encrypted **Votes** are multiplied the obtained result **E** (computed mod N^2) can be correctly decrypted and indicate the sum of all **Votes**.
Then encoding of **Votes** for every candidate must be chosen in such a way that they can be distinguished from the sum of Votes of other candidate.

Let us consider three candidates **Can1**, **Can2**, **Can3** for our generated **PuK**=**N**=13843, |N|=14 bits.
For **Votes** separation of 3 **Candidates** we assign the total sum of **Votes** represented by 4 bits.
This sum can be achieved by optimal encoding of **Votes** consisting of the following cases.

1. The **Vote** for **Can1** is encoded by number 2^8=**256**. Then If all 15 **Voters** vote for **Can1** the total sum of **votes** will be 15∗256=3840. Notice that 3840+256=4096=2^{12}.
2. The **Vote** for **Can2** is encoded by number 2^4=**16**. If all 15 **Voters** vote for **Can2** the total sum will be 15∗16=240. Notice that 240+16=256=2^8.
3. The **Vote** for **Can3** is encoded by number **1**. If all 15 **Voters** vote for **Can1** the total sum will be 15.

Then the maximal sum of votes is obtained in the case 1 and is equal to 3840 < 14351=**N**.
In tables below the maximal sum of Votes for **Can1**, **Can2**, **Can3** encoded in binary with 4 bit length is presented.
Then the maximal sum of **Voters** can not exceed number $15=2^4-1=1111_b$.

0	0	0	0	0	0	0	0	0	0	0	1
Can1				Can2				Can3			

For **Can1**: $0000\ 0000\ 0001_b=1$

0	0	0	0	0	0	0	1	0	0	0	0
Can1				Can2				Can3			

For **Can2**: $0000\ 0001\ 0000_b=2^4=16$

0	0	0	1	0	0	0	0	0	0	0	0
Can1				Can2				Can3			

For **Can3**: $0001\ 0000\ 0000_b=2^8=256$

0	0	0	0	0	0	0	0	1	1	1	1
Can1				Can2				Can3			

Sum of total votes for every candidate:

For **Can3**: 0000 0000 1111_b=15

0	0	0	0	1	1	1	1	0	0	0	0
Can1				Can2				Can3			

For **Can2**: 0000 1111 0000_b=240

1	1	1	1	0	0	0	0	0	0	0	0
Can1				Can2				Can3			

For **Can1**: 1111 0000 0000_b=3840

The Globe wide Voting

Let us imagine that election is performed in the half of the Globe with number of **Voters M** is about 4 billions.
Let **M** < 2^{32} = 4 294 967 296.
Let the number of **Candidates** to be elected is about 1000.
Let **K** < 2^{10} = 1 024.
Then the number of bits for election data representation for every of 1024 = 2^{10} **Candidates** is
$2^{10} * 2^{32} = 2^{42}$ = 4 398 046 511 104 and is about 4 trillions.
Then the maximal sum of **Votes** is **K**∗**M** and is represented by 2^{42} = 4 398 046 511 104 bits number and is corresponding to the decimal number (2)^(2^{42}) - 1 = $2^{4\,398\,046\,511\,104}$ - 1.
Since the sum of **Votes** must be less than **PuK**=**N**, then **N** must be close to the number $2^{4\,398\,046\,511\,104}$ - 1.
Then |**N**|= 4 398 046 511 104 bits.
Since **N**=**p**∗**q**, where **p**, **q** are primes, then |**p**|=|**q**|= 2 199 023 255 552 bits.
The problem is to generate such a big prime numbers.
If we encode decimal numbers in ASCII code then 1 decimal digit is encoded by 8 bits.
Then **p**, **q** numbers in decimal representation will have 2 199 023 255 552 / 8 = 274 877 906 944 decimal digits.
It is more than 274 billions.

Problem solution.

The solution is to divide election into different **Voting Areas** so reducing number of **Voters M**.
Then encryption scheme becomes more practical and more efficient realizable.
Let we are able to generate considerable large prime numbers **p**, **q** having 2^{15} = 32 768 bits,
i.e. |**p**|=|**q**|= 2^{15} = 32 768 bits and hence are bounded by 2^{32768} - 1 such a huge decimal number.
Notice that in traditional cryptography for prime numbers it is enough to have 4096 bit length.
Then **N**=**p**∗**q** will have 32 768 + 32 768 = 65 536 = 2^{16} bit length and hence is bounded by the following $2^{65\,536}$ - 1 huge decimal number.
Then the arithmetic operations are performed with such a huge numbers and even with numbers up to **N^2** since operations **mod** **N^2** are used. Therefore the special software is needed.
Let **Voting Areas** are divided in such a way that they can serve about 16 millions **Voters**.
Assume that number of **Voters M** < 16 777 215 = 2^{24} - 1. Then |**M**| = 24 bits.
Then for every candidate we must dedicate 24 bits in the total string of bits of number **PuK**=**N** where |**N**|=2^{16} = 65 536.
Then number of **Candidates K** in **Voting Area** is the following:

K = |**N**| / |**M**| = 2^{16} / 24 = 2731.

The distribution of **Candidates** and the number of bits them assigned is presented in table.

Total length of **N** is 65 536 bits

24 bits	24 bits	24 bits		24 bits
Can1	**Can2**	**Can3**	⟵⟶	**Can2731**

There are 2 problems must be solved:
1. To generate 2 large prime numbers **p**, **q** having 2^{15} = 32 768 bit length ~ 10^{10000} : it is feasible.
2. To perform a computations with large numbers using special software having 2^{32} = 4 294 967 296 bits when operations are performed **mod N²**.

```
>>  p=109;
>> q=127;
>> N=p*q                                   % PuK=N=13843
N = 13843
                                          % |N|=14 bits
>> N_2=int64(N*N)
N_2 = 191628649
>> dec2bin(N_2)
ans = 1011 0110 1100 0000 0101 0110 1001

>> fy=(p-1)*(q-1)
fy = 13608                                % PrK=fy
```

- **Enc**: on input a public key N and a message $m \in \mathbb{Z}_N$, choose a random $r \leftarrow \mathbb{Z}_N^*$ and output the ciphertext

$$c := [\underbrace{(1+N)^m}_{e_1 \bmod N^2} \cdot \underbrace{r^N}_{e_2 \bmod N^2} \bmod N^2].$$

$$c = e = e_1 \cdot e_2 \bmod N^2$$

$\mathbb{Z}_N^* = \{z \mid gcd(z,N)=1\}$

$z < N-1$

```
>> vw=16
vw = 16
>> rw=randi(N-1)
rw = 5029
>> gcd(rw,N)
ans = 1
>> e1=mod_exp((1+N),vw,N_2)
e1 = 221489
>> e2=mod_exp(rw,N,N_2)
e2 = 115257872
>> ew=mod(e1*e2,N_2)
ew = 157077575
```

```
>> w2=256
v2 = 256
>> r2=randi(N-1)
r2 = 12539
>> gcd(r2,N)
ans = 1
>> e21=mod_exp((1+N),v2,N_2)
e21 = 3543809
>>  e22=mod_exp(r2,N,N_2)
e22 = 57431777
>> ee2=mod(e21*e22,N_2)
ee2 = 184773534
```

```
>> E=mod(ew*ee2,N_2)
E = 108508702
```

$$c^{\phi} \bmod N^2 = d_1$$

$$\phi^{-1} \bmod N = d_3$$

$$m := \left[\underbrace{\frac{[c^{\phi(N)} \bmod N^2] - 1}{N}}_{} \cdot \underbrace{\phi(N)^{-1}}_{} \bmod N \right].$$

$$m = d_2 \cdot d_3 \bmod N$$

$$\frac{d_1 - 1}{N} \bmod N = d_2$$

```
>> d1=mod_exp(E,fy,N_2)
d1 = 73298686
>> d2-mod((d1-1)/N,N)
ans = 8462
>> d2=mod((d1-1)/N,N)
d2 = 5295
>> fy_m1=mulinv(fy,N)
fy_m1 = 1885
>> d3=fy_m1
d3 = 1885
>> m=mod(d2*d3,N)
m = 272
>> V=m
V = 272
```

Can1 := 256

```
>> NVCan1=floor(272/256)
NVCan1 = 1
>> 272/256
ans = 1.0625
```

Can2 := 16

```
>> vv=V-1*256
vv = 16
```

Can3 := 1

P.Vardas	No	ri	ci	c	V_by_M: cMi	c*cMi	Dec(c*cMi)	Tot_S_of_V
Au. Juozas	1	16339	149318501	216987098	92831661	152067656	896	896
Be. Antanas	2	8609	32143614	216987098	123083220	203234256	896	896
	3							
	4							
	5							
	6							
	7							
	8							
	9							
	10							
	11							
	12							
	12							
	14							
	15							

ri		Random number generated for Paillier encryption
ci		Your vote encrypted by Paillier encryption
c		The product of all encrypted votes in your polling station. **Provided by lecturer**

V_by_M: cMi	Encrypted Vote received by Mail: cMi. **Provided by lecturer**
c*cMi	Multiplied encrypted votes in polling station multiplied by cMi
Dec(c*cMi)	Decryption all multiplied votes
Tot_S_of_V	Total sum of votes

No	N_of_V_Can1	N_of_V_Can2	N_of_V_Can3	Tot_N_of_V	Dec(cMi)	**Acc/Dec** cMi	Can1	Can2	Can3
1	5	8	0	13	512, 2 balsai uz pirma	Dec	3	8	0
2	6	8	0	14	256, 256, 256	Dec	3	8	0
3									
4									
5									
6									
7									
8									
9									
10									
11									
12									
12									
14									
15									

N_of_V_Can1	Number of votes for Can1
N_of_V_Can2	Number of votes for Can2
N_of_V_Can3	Number of votes for Can3
Tot_N_of_V	Total number of votes
Dec(c*Mi)	Decrypted vote cMi received by Mail
Acc/Dec cMi	Accept or Decline vote received by Mail. Input: **Acc** or **Dec**
Can1	Number of votes for Can1
Can2	Number of votes for Can2
Can3	Number of votes for Can3

Till this place

ElGamal Cryptosystem

$PP = (p_E, g_E)$

$\ldots = (p_E, g_E)$

Voters $\{V_i\}_1^M$

$PuK_{EC} = N.$

$PrK = x$; $PuK = a$.

$x \leftarrow randi(\ -1)$

$a = g_E^x \bmod p_E$

TTP

$y \leftarrow randi(p_E - 1)$

$b = g_E^x \bmod p_E$

$PrK_{TTP} = y$, $PuK_{TTP} = b$;

$PuK_{EC} = N.$

$\{a_1, a_2, \ldots, a_M\}$

$Cert_1$

EC: Paillier scheme

$PuK_{EC} = N$, $PrK_{EC} = \phi$;

$N = \underbrace{p \cdot q}$; $\leftarrow$ primes

votes $\{v_i\}_1^M$

$V_1 \leftarrow PrK_1 = x_1$, $PuK_1 = a_1$;

$Enc(PuK_{EC}, v_1) = c_1$

$TS_1 \leftarrow$ Time stamp Authority

$h_1 = H(a_1 || c_1 || TS_1)$

$Sign(PrK_1, h_1) = \sigma_1$

$\dfrac{a_1, c_1}{TS_1, \sigma_1}$

$Verf(Cert_1)$

$Verf(a_1)$

$Verf(\sigma_1)$

$Verf(TS_1)$

— — — — — — —

V_{255} - - - - -

$\dfrac{a_{255}, c_{255}}{TS_{255}, \sigma_{255}}$

TTP

Block B_1

Data D_1:

$TS_1, \ldots, TS_{255}$

$c_1, \ldots, c_{255}$

$C_{B1} = \prod_{i=1}^{255} c_i \bmod N^2$

$h_{B1} = H(D_1)$

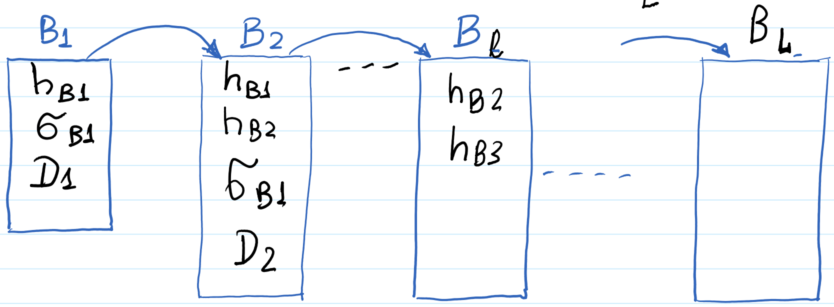
Let number of Voter Areas is L

$$\boxed{h_{B1} = H(D_1)}$$

$$Sign(y, h_{B1}) = \sigma_{B1}$$

Block B_ℓ

Block B_L



2^0: 0000 0001

2^8: 0000 0001 0000 0000

2^{16}: 0000 0001 0000 0000 0000 0000

2^{24}: 0000 0001 0000 0000 0000 0000 0000 0000

1) $|p| = |q| = 1024$ bits

$N = p \cdot q \rightarrow |N| = 2048$ bits

$2048 : 8 = 256$ candidates

$\begin{array}{l} 2048 \\ \underline{16} \\ \ \ 44 \\ \ \ \underline{40} \\ \ \ \ \ 48 \\ \ \ \ \ \underline{48} \end{array}$

2) >> 2^16
ans = 65536

$|p| = |q| = 2^{16} = 65536$ bits

$N = p \cdot q \rightarrow |N| = 2^{32}$ bits

>> 2^32
ans = 4 294 967 296
>> ans/256
ans = 16 777 216

$4\,294\,967\,296 : 2^8 =$

$= 16\,777\,216$

CONSTRUCTION 11.32

Let GenModulus be a polynomial-time algorithm that, on input 1^n, outputs (N, p, q) where $N = pq$ and p and q are n-bit primes (except with probability negligible in n). Define a public-key encryption scheme as follows:

- **Gen**: on input 1^n run GenModulus(1^n) to obtain (N, p, q). The public key is N, and the private key is $\langle N, \phi(N)\rangle$.
- **Enc**: on input a public key N and a message $m \in \mathbb{Z}_N$, choose a random $r \leftarrow \mathbb{Z}_N^*$ and output the ciphertext

$$c := [(1+N)^m \cdot r^N \bmod N^2].$$

- **Dec**: on input a private key $\langle N, \phi(N)\rangle$ and a ciphertext c, compute

$$m := \left[\frac{[c^{\phi(N)} \bmod N^2] - 1}{N} \cdot \phi(N)^{-1} \bmod N\right].$$

The Paillier encryption scheme.

$|N^2|$=28 bits
$|N|$= 14 bits
$|p|$ = 7 bits
$|q|$ =7 bits

EC key generation
p = 127
q = 113
n = 14351

$\downarrow 2^0$
Aldona = 0000000 0000001
Bronius = 0000001 0000000
$\uparrow 2^7$

$r \leftarrow$ randi

$$C_{mm} = (1+N)^V \cdot r^N \bmod N^2$$

$$C_{mm\,l} = (1+N)^V \bmod N^2\ ; \quad C_{mm\,r} = r^N \bmod N^2 ;$$

$$C_{mm} = C_{mm\,l} \cdot C_{mm\,r} \bmod N^2$$

```
>> v1=2^7
v1 =  128
>> n_2=n*n
n_2 =  205951201
>> cl=mod_exp((1+n),v1,n_2)
cl = 1836929
>> r1=randi(4783)
r1 =  208
>> cr=mod_exp(r1,n,n_2)
cr = 19896092
>> c1=mod(cl*cr,n_2)
c1 = 20154410
```

```
c1 = 20154410
c13 = 50747781
c35 = 18351792
c44 = 66165185

>> cB1=c1*c13*c35*c44
cB1 = 9223372036854775807
>> cB1mn_2=mod(cB1,n_2)
cB1mn_2 = 137726674
>> cB1=cB1mn_2
cB1 = 137726674
```

```
>> hB1=hd28('c1=20154410||c13=50747781||c35=18351792||c44=66165185||cB1=137726674')
hB1 =  110980798
```

CONSTRUCTION 11.32

Let GenModulus be a polynomial-time algorithm that, on input 1^n, outputs (N, p, q) where $N = pq$ and p and q are n-bit primes (except with probability negligible in n). Define a public-key encryption scheme as follows:

- **Gen**: on input 1^n run GenModulus(1^n) to obtain (N, p, q). The public key is N, and the private key is $\langle N, \phi(N)\rangle$.
- **Enc**: on input a public key N and a message $m \in \mathbb{Z}_N$, choose a random $r \leftarrow \mathbb{Z}_N^*$ and output the ciphertext

$$c := [(1+N)^m \cdot r^N \bmod N^2].$$

- **Dec**: on input a private key $\langle N, \phi(N)\rangle$ and a ciphertext c, compute

$$m_1 = c^{\phi} \bmod N^2$$

$$m_2 = (m_1 - 1)/N \bmod N$$

$$m_3 = \phi^{-1} \bmod N$$

$$V_{\Sigma} = m_2 \cdot m_3 \bmod N$$

cB1 = 137726674

random $r \leftarrow \mathbb{Z}_N^*$ and output the ciphertext

$$c := [(1+N)^m \cdot r^N \bmod N^2].$$

- Dec: on input a private key $\langle N, \phi(N) \rangle$ and a ciphertext c, compute

$$m := \left[\frac{[c^{\phi(N)} \bmod N^2] - 1}{N} \cdot \phi(N)^{-1} \bmod N \right].$$

The Paillier encryption scheme.

$v_\Sigma = m_2 \cdot m_3 \bmod N$

```
cB1 = 137726674
>> fy
fy =  14112
>> m1=mod_exp(cB1,fy,n_2)
m1 = 4677410
>> m2=mod((m1-1)/n,n)
m2 = 326
>> gcd(fy,n)
ans =  1
>> m3=mulinv(fy,n)
m3 = 5224
>> vsigma=mod(m2*m3,n)
vsigma = 9606
>> vsigma/127
ans = 76
```