```
>> p=15
p = 15
>> p=15;
>> a=38;
>> am15=mod(a,p)
am15 = 8
>> b=41
b = 41
>> apbmp=mod(a+b,p)
apbmp = 4
>>
>> ambmp=mod(a-b,p)
ambmp = 12
>> amubmp=mod(a*b,p)
amubmp = 13
```

Wrong computations

```
>> adbmp=mod(a/b,p)
adbmp = 0.9268
>>
>> adbmp=int64(mod(a/b,p))
adbmp = 1      % needs a verification
>> gcd(b,p)
ans = 1
```

```
> n=15;
>> a=7;
>> b=5;
>> b_m1=mulinv(b,n)
b_m1 = Inverse element does not exist
>> gcd(b,n)     % since gcd(b,n)=/=1
ans = 5
>> c=11;
>> c_m1=mulinv(c,n)     % finding an inverse element to c mod n
c_m1 = 11
>> cmc_m1=mod(c*c_m1,n) % verification of inv. element
cmc_m1 = 1
>> gcd(c,n)     % inverse element exists since gcd(c,n)=1
ans = 1
```

Computations mod p when p is prime

```
>> p=11;
>> a
a = 7
>> b
b = 5
>> b_m1=mulinv(b,p)
b_m1 = 9
>> check=mod(b*b_m1,p)
check = 1

>> amb_m1=mod(a*b_m1,p)
amb_m1 = 8
```

Random numbers generation

```
>> r=randi(2^28-1)
r = 3.7044e+07
>> r=int64(randi(2^28-1))
r = 193575539
>> rr=int64(randi(2^28-1))
rr = 28366443
>> rb=dec2bin(r)
rb = 1011 1000 1001 1011 1010 0111 0011
>> rrb=dec2bin(rr)
rrb = 1101100001101011001101011
>> rh=dec2hex(r)
rh = B89BA73
>> check=hex2dec(rh)
check = 1.9358e+08
>> check=int64(hex2dec(rh))
check = 193575539
```

Primes and strong primes generation

Finding a generator g

Primes and strong primes generation
having 28 bit length
>> p=genprime(28)
p = 178892729
>> pb=dec2bin(p)
pb = 10101010100110101111110111001
>> p=genstrongprime(28)
p = 207101879 % this p is strong prime: p=**2***q+1
% this p defines a set Z_p^*= {1, 2, 3, …,207101878}
>> pb=dec2bin(p)
pb = 11000101100000011111110110111

Finding a generator g
Generator g is found using a modular exponent function
$z=b^e \bmod p$ <--> >> z=mod_exp(b,e,p)
Where b is in the set Z_p^* and e is in the set Z_{p-1}
Z_{p-1}={0, 1, 2, 3, …, p-1} wher +, -, *, operations are defined mod p-1.

>> b=12345
b = 12345
>> e=56789
e = 56789
>> z=mod_exp(b,e,p)
z = 110294140

Power Tab. Z_{11}* mod 11											
^	0	1	2	3	4	5	6	7	8	9	10
1	1	1	1	1	1	1	1	1	1	1	1
2	1	2	4	8	5	10	9	7	3	6	1
3	1	3	9	5	4	1	3	9	5	4	1
4	1	4	5	9	3	1	4	5	9	3	1
5	1	5	3	4	9	1	5	3	4	9	1
6	1	6	3	7	9	10	5	8	4	2	1
7	1	7	5	2	3	10	4	6	9	8	1
8	1	8	9	6	4	10	3	2	5	7	1
9	1	9	4	3	5	1	9	4	3	5	1
10	1	10	1	10	1	10	1	10	1	10	1

Z_{11}* = {1, 2, 3, …,10} 10=p-1
In general
Z_p* = {1, 2 , 3, …, p-1}
The exponents are in the set
Z_{p-1} = {0, 1, 2 , 3, …, p-2} with operations **+, -, *** (and in certain cases **/** when gcd(z, p-1)=1 and z is in Z_{p-1}) are performed modulo p-1.
In the case of p=11, Z_{p-1} = Z_{10} = {0, 1, 2, 3, …, 9}

To see the unique exponents in Z_{10} compute
>> g=2;
>> p=11;
>> g_10 = mod_exp(g,p,10)
>> g_11 = mod_exp(g,p,11)
>> g_12 = mod_exp(g,p,12)
>> g_13 = mod_exp(g,p,13)
>> g_14 = mod_exp(g,p,14)
>> …..

Attention! If p - is prime, exponent p-1 yelds the same result as exponent 0, e.g. exponent p-1 is equivalent ↔ to exponent 0.
Then any exponent can be reduced modulo p-1 that saves the computation resources.
It is a consequence of Fernat little theorem.
$10 \bmod 10 = 0 \quad \rightarrow \quad g^{10} \bmod 10 = g^0 \bmod 10$
$11 \bmod 10 = 1 \quad \rightarrow \quad g^{11} \bmod 10 = g^1 \bmod 10$
$12 \bmod 10 = 2 \quad \rightarrow \quad g^{12} \bmod 10 = g^2 \bmod 10$
…..
In general: for any integer s and p - prime: $g^s \bmod p = g^{s \bmod (p-1)} \bmod p$.

For example let s = r + xh
Then before g powering by z it is required to perform reduction: s = r + xh mod p-1
And then compute: g^s mod p which is, for example, a left side of signature verification equation will be presented later.

If ***p*** is **strong prime** if ***p***=2***q*** + 1 where ***q*** = (***p***-1)/2 is prime as well.
Then ***g*** in Z_P* is a generator of Z_P* if and only if
(**iff**) $g^2 \neq 1 \bmod p$ **and** $g^q \neq 1 \bmod p$.

> g=2;
>> first_c=mod_exp(g,2,p)
first c = 4

If p is **strong prime** if $p=2q+1$ where $q=(p-1)/2$ is prime as well.
Then g in Z_P^* is a generator of Z_P^* if and only if
(**iff**) $g^2 \neq 1 \bmod p$ **and** $g^q \neq 1 \bmod p$.

```
>> q=(p-1)/2
q = 103550939
>> g=2;
……
```

Founded generator is

```
>> g=123;
```

Private and public keys computation.

1. Private key generation

```
>> x=randi(2*28-1)
x = 30
>> x=randi(2^28-1)
x = 2.0122e+08     % Result must be integer
>> x=int64(randi(2^28-1))
x = 247270535
```

1. Public key computation

```
>> a=mod_exp(g,x,p)
a = 131503982
```

```
> g=2;
>> first_c=mod_exp(g,2,p)
first_c = 4
>> second_c=mod_exp(g,q,p)
second_c = 1
>> g=3;
>> second_c=mod_exp(g,q,p)
second_c = 1
>> g=4;
>> second_c=mod_exp(g,q,p)
second_c = 1
>> g=5;
>> second_c=mod_exp(g,q,p)
second_c = 1
>> g=6;
>> second_c=mod_exp(g,q,p)
second_c = 1
>> g=123;
>> second_c=mod_exp(g,q,p)
second_c = 207101878
```