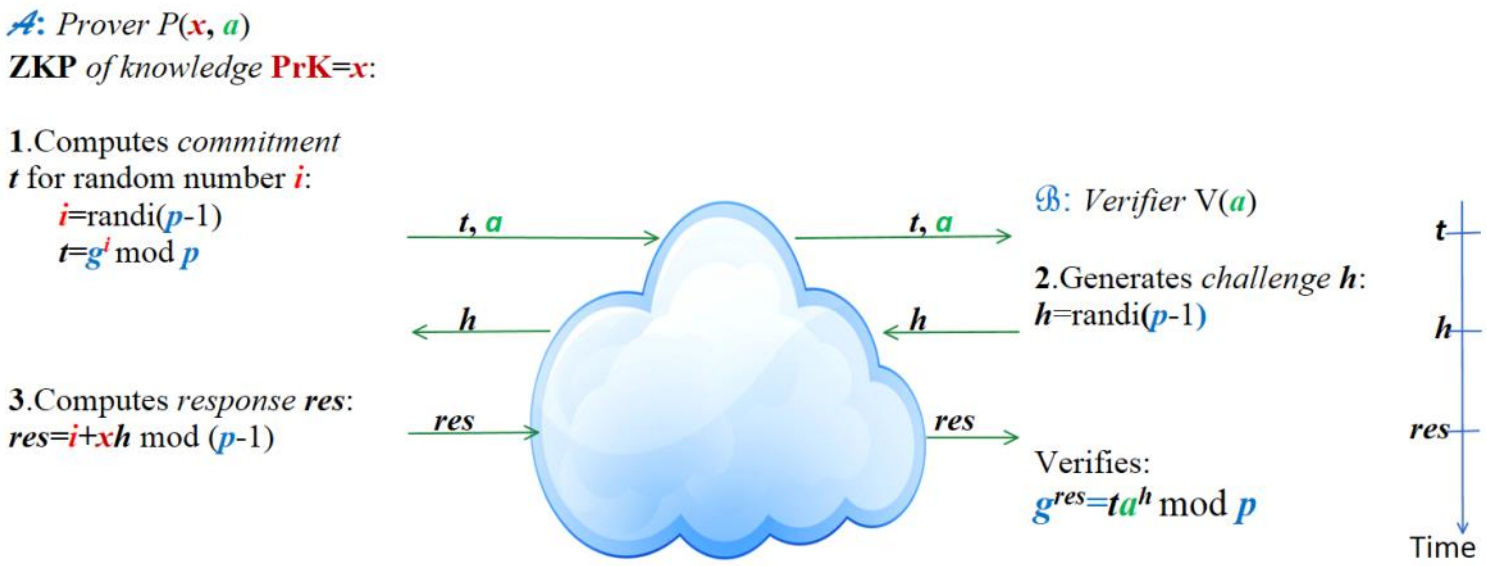
PP = (***p***, ***g***).

Schnorr Id is interactive protocol, but not recurent as it realized to prove the mirckle words.
Schnorr Id Scenario: **Alice** wants to prove **Bank** that she knows her Private Key - $PrK_A = x$ which corresponds to her Public Key - $PuK_A = a = g^x \bmod p$ not revealing $PrK_A = x$.



Correctness:
$g^{res} \bmod p = g^{i+xh \bmod (p-1)} \bmod p = g^i g^{xh} \bmod p = t(g^x)^h \bmod p = ta^h \bmod p$.

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

>> x=int64(randi(p-1))
x = 130007868
>> a=mod_exp(g,x,p)
a = 176162386
```

```
>> i=int64(randi(p-1))
i = 65140707

>> t=mod_exp(g,i,p)
t = 189627349
>> h=int64(randi(p-1))
h = 242654365
>> res=mod(i+x*h,p-1)
res = 20761443
```

```
>> res=mod(i+x*h,p-1)
res = 20761443
>> g_res=mod_exp(g,res,p)
g_res = 215652516
>> V1=g_res
V1 = 215652516
>> a_h=mod_exp(a,h,p)
a_h = 73404223
>> ta_h=mod(t*a_h,p)
ta_h = 215652516
>> V2=ta_h
V2 = 215652516
```

Alice chooses at random ***u***, 1<***u***<***p***-1 and computes first component ***r*** of his signature:

$$r = g^u \bmod p. \qquad (2.19)$$

Alice computes H-function value ***h*** and second component ***s*** of her signature:

$$h = H(M \| r), \qquad (2.20)$$

$$s = u + xh \bmod (p-1). \qquad (2.21)$$

Alice's signature on ***h*** is **σ=(*r*, *s*)**. Then **Alice** sends ***M*** and **σ** to **Bob**.

After receiving M' and σ, **Bob** according to (2.20) computes h'

$$h'=H(M'||r),$$

and verifies if

$$g^s \bmod p = r a^{h'} \bmod p. \qquad (2.22)$$

V1 V2

Symbolically this verification function we denote by

$$Ver(a,\sigma,h')=V\in\{True, False\}\equiv\{1, 0\}. \qquad (2.23)$$

This function yields *True* if (2.22) is valid if: $h=h'$ and $PuK_A = a = F(PrK_A) = g^x \bmod p$.

and: $M=M'$

```
>> u=int64(randi(p-1))
u = 158865637
>>  r=mod_exp(g,u,p)
r = 44519947
>> cc=concat(m,r)
cc = Hello Bob44519947
>> h=hd28(cc)
h = 27628438
>> s=mod((u+x*h),p-1)
s = 267483811
```

```
>> g_s=mod_exp(g,s,p)
g_s = 239508680
>> V1=g_s
V1 = 239508680
>> a_h=mod_exp(a,h,p)
a_h = 128538175
>> ra_h=mod(r*a_h,p)
ra_h = 239508680
>> v2=ra_h
v2 = 239508680
```

```
>> m='Hello Bob'
m = Hello Bob
>> u=int64(randi(p-1))
u = 228451192
>> r=mod_exp(g,u,p)
r = 33418907
>> cc=concat(m,r)
cc = Hello Bob33418907      % cc is a string type variable
>> cc=concat(m,'33418907')
cc = Hello Bob33418907
>> cc=concat(m,'r')
cc = Hello Bobr
```

```
>> h=hd28(cc)
h = 104824510      104824510
>>  s=mod((u+x*h),p-1)
s = 147250342
```

```
>> g_s=mod_exp(g,s,p)
g_s = 185672370
V1=g_s;
>> a_h=mod_exp(a,h,p)
a_h = 263774143
>>  V2=mod(r*a_h,p)
V2 = 185672370
```

RSA Blind signature

```
>> p=genprime(14)
p = 10069
>> dec2bin(p)
ans = 10011101010101
>> q=genprime(14)
q = 15541
>> dec2bin(q)
ans = 11110010110101
```

```
>> fy=(p-1)*(q-1)
fy = 156456720
>> d=mulinv(e,fy)
d = 105260993
>> mod(e*d,fy)
ans = 1
>> m=100
m = 100
```

q = 13541
>> dec2bin(q)
ans = 11110010110101
>> n=p*q
n = 156482329
>> dec2bin(q)
ans = 11110010110101
>> dec2bin(n)
ans = 10010101001110111011000110 01
>> e=2^16+1
e = 65537

ans = 1
>> m=100
m = 100
>> t=int64(randi(n))
t = 61584152
>> gcd(t,n)
ans = 1

>> t_e=mod_exp(t,e,n)
t_e = 150571433
>> mm=mod(m*t_e,n)
mm = 34839716

$m' = m \cdot t^e \bmod n$ $\xrightarrow{mm \equiv m'}$

$Ver(Puk=(n,e), \sigma', m') = m'$ $\xleftarrow{\sigma'}$

B:

$Sign(Prk=d, m') = \sigma'$

$\sigma' = (m')^d \bmod n =$

$= (m \cdot t^e)^d \bmod n =$

$= m^d \cdot t^{ed} \bmod n \xrightarrow{ed \bmod \phi = 1} \bmod n =$

$= m^d \cdot t \bmod n$

$\xleftarrow{\sigma' = m^d \cdot t \bmod n}$

>> SigB=mod_exp(mm,d,n)
SigB = 76295822
>>
>> mmm=mod_exp(SigB,e,n)
mmm = 34839716

A: unmasks signed m'

$(\sigma')^e \bmod n = ((m')^d)^e \bmod n = (m')^{ed \bmod \phi} \bmod n =$ ($ed \bmod \phi = 1$)

$= m' \bmod n = m' \Rightarrow$ Signature is valid. = True

if $m' < n$

>> mmm=mod_exp(SigB,e,n)
mmm = 34839716

A extracts (unmasks) $m^d \bmod n = \sigma$ from σ':

$\sigma' \cdot t^{-1} \bmod n$ → if $gcd(t, n) = 1 \Rightarrow t^{-1} \bmod n$ exists.

$\sigma' \cdot t^{-1} \bmod n = m^d \cdot t \cdot t^{-1} \bmod n = m^d \bmod n = \sigma.$

>> SigBm=mod(SigB*t_m1,n)
SigBm = 143112175
>> m100=mod_exp(SigBm,e,n)
m100 = 100