Diffie-Hellman Key Agreement Protocol (DH KAP)

Public Parameters PP=(p,g)

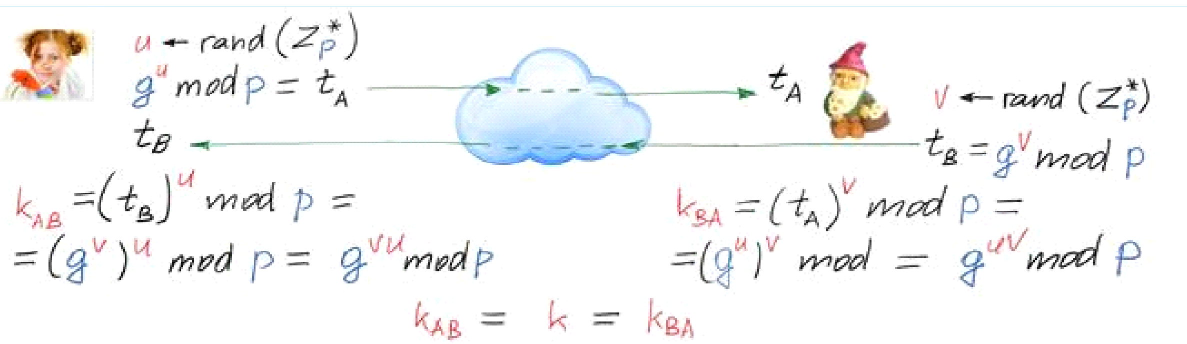
In general, it is a hard problem, but using strong prime p and *Lagrange theorem in group theory* the generator in Z_p^* can be found by random search satisfying two following conditions if p is strongprime: $p = 2*q + 1 \dashrightarrow q = (p-1)/2$

For all $g \in \Gamma$

$$g^q \neq 1 \ \mathbf{mod}\ p; \text{ and } g^2 \neq 1 \ \mathbf{mod}\ p.$$

```
>> p=genstrongprime(28)
p = 224013599
>> q=(p-1)/2
q = 112006799
```

```
>> g=111;
>> mod_exp(g,q,p)
ans = 224013598
>> u=int64(randi(2^28-1))
```




```
>> u=int64(randi(2^28-1))
u = 195162450
>> dec2bin(u)
ans = 1011 1010 0001 1111 0001 0101 0010
>> tA=mod_exp(g,u,p)
tA = 22053505
>> kAB=mod_exp(tB,u,p)
kAB = 196960461
```

```
>> v=int64(randi(2^28-1))
v = 212879876
>> tB=mod_exp(g,v,p)
tB = 179573345
>> kBA=mod_exp(tA,v,p)
kBA = 196960461
```

```
>> M='Hello Bob'
M = Hello Bob

>> k=kBA
k = 196960461
>> kh32=dec2hex(k,32)
kh32 = 000000000000000000000000BBD60CD
>> NR=1;
>> fun='e'
fun = e
>> in=M
in = Hello Bob
>> AES128(in,kh32,NR,fun)
new =   R  H6y  N
ans = 18 b3ed521880 cb4836c6de 79810ebe4e
```

```
% AES128(in,kh32,NR,fun) Advanced Encryption Standard symmetric cipher with key length of 128 bits
%                    Encryption is performed for 1 block of length 128 bits or 16 ASCII symbols
%
% in - plaintext/ciphertext of string type: maximum 16 symbols or shorter
%
% kh32 - shared secret key in hexadecimal number of length=32 (128 bits)
% kh32 can be obtained when shared decimal key k is given using commands:
%      >> k=int64(randi(2^28))
%      k = 160966896
%      >> kh32=dec2hex(k,32)
%      kh32 = 000000000000000000000000099828F0
%
% NR - Number of Rounds (e.g. NR = 10)
%     The smaller NR, the lower security of encryption but the speed of encryption is higher
%     The least number of NR is 1 and in this case security lack is evident
%
```

```
>> AES128(in,kh32,NR,fun)
new =   R  H6y  N
ans = 18 b3ed521880 cb4836c6de 79810ebe4e

>> Ch=ans
Ch = 18b3ed521880cb4836c6de79810ebe4e
>>
>> fun='d'
fun = d
>> Dh = AES128(Ch,kh32,NR,'d')
Out = 0000000000000048656c6c6f20426f62
Dh = Hello Bob
```

Savarankiškai atlikite šifravimą su saugiu N >= 10.

```
%     The smaller NR, the lower security of encryption but the speed of encryption is higher
%     The least number of NR is 1 and in this case security lack is evident
%
% fun - letter determining either encription: fun='e' or decryption: fun='d' functions
%
% Encryption example:
% >> in = 'Hello Bob';
% >> kh32 = '000000000000000000000000099828F0';
% >> NR = 10;
% >> Ch = AES128(in,kh32,NR,'e')
% ASCII_e =   ?1  ~mV               % ciphertext in ASCII format
% Ch = 0f9a2c08d191310fb27ed16d90f45686   % ciphertext in hexadecimal format
%
% Decryption example:
% >> Dh = AES128(Ch,kh32,NR,'d')
% Dh = 00000000000048656c6c6f7720426f62   % decrypted message in hex format
% D = Hello Bob                         % Decrypted message in ASCII format
```

Public Parameters PP=(p,g)

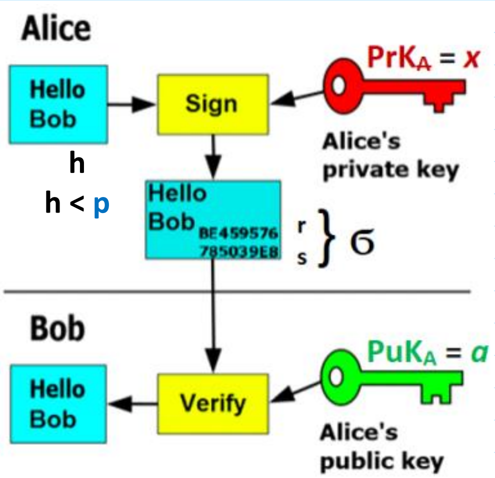
```
>> p
p = 224013599
>> g
g = 111
```

PrK = x <-- randi ==> PuK = a = g^x mod p

```
% input - string array, i.e "asd"
% output - 28 LSBs in hexdecimal form of SHA-256
% >> h=h28('Hello Bob')
h = BD9003E

% input - string array, i.e "asd"
% output - 28 LSBs in decimal form of SHA-256
% >> h=hd28('Hello Bob')
h = 198770750
```

```
>> x=int64(randi(2^28-1))
x = 121821633
>> a=mod_exp(g,x,p)
a = 192863004
```



Signature creation for message M >> p.

1. Compute decimal h-value $h=H(M)$; $h<p$.
2. Generate >> i =int64(randi(p-1)) % such that gcd(i,p-1)=1.
3. Compute i^{-1} mod (p-1). You can use the function
   >> i_m1=mulinv(i, p-1);
4. Compute $r=g^i$ **mod** p.
5. Compute $s=(h-xr)i^{-1}$ mod (p-1).
6. Signature on h-value h is σ = (r,s)
   Sign(x,h) = σ = (r,s).

```
>> i =int64(randi(p-1))
i = 202651315
>> gcd(i,p-1)
ans = 1
>>  i_m1=mulinv(i, p-1)
i_m1 = 59189655
>> mod(i*i_m1,p-1)
```

$A: \xrightarrow{M,\ \sigma=(r,s)} B:$

```
>> h = hd28('Hello Bob')

>> V1 = mod_exp(g,h,p)
```

```
ans = 1
>>  i_m1=mulinv(i, p-1)
i_m1 = 59189655
>> mod(i*i_m1,p-1)
ans = 1
> r=mod_exp(g,i,p)
r = 136585182
>> hmxr=mod(h-x*r,p-1)
hmxr = 76861196
>> s=mod(hmxr*i_m1,p)
s = 106387113
```

```
>> h = hd28('Hello Bob')

>> V1 = mod_exp(g,h,p)

>> a_r = mod_exp(a,r,p)

>> r_s = mod_exp(r,s,p)

>> V2 = mod(a_r*r_s,p)
```

2.***Signature Verification***

A signature σ=(***r***,***s***) on message ***M*** is verified using Public Parameters **PP**=(**p**, **g**) and $\mathbf{PuK_A}$=**a**.

1. Bob computes **h=H(M)**.
2. Bob verifies if 1<**r**<**p-1** and 1<**s**<**p-1.**
3. Bob calculates $\mathbf{V1=g^h}$ **mod p** and $\mathbf{V2=a^r r^s}$ **mod p,** and verifies if **V1=V2.**

The verifier Bob accepts a signature if all **conditions** are satisfied during the signature creation and rejects it otherwise.