During the exam you must solve 5 problems from
https://imimsociety.net/en/14-cryptography

Cryptography

Information: **Confidentiality** | **Integrity and Authenticity** | **Person Identification**

Encryption/Decryption	Signing/Verification	Schnorr Identification
ElGamal Encryption	Schnorr Signature	Non-Interactive Zero Knowledge Proof (NIZKP)
	ElGamal Signature	

Public Parameters PP = (p, g).
p = 268435019; g=2;

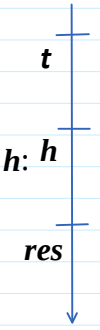
Schnorr Identification is an interactive Zero Knowledge Proof - ZKP

$\mathcal{A}$: ZKP of knowledge x:
$PrK_A = x$ = randi(p-1)
$PuK_A = a = g^x \bmod p$
1.Computes **commitment** t for random number i:
i=randi(p-1)
$t=g^i \bmod p$
3.Computes **response** *res*:
$res=i+xh \bmod (p-1)$

t, a →
← h
res →

t, a →
← h
res →

$\mathcal{B}$: $PuK_A = a$
2.Generates **challenge** h: h=randi(p-1)
Verifies:
$g^{res}=ta^h \bmod p$



Alice: 'Hello Bob'
>> M='Hello Bob'
M; $\sigma=(r,s)$; a →

M'; $\sigma=(r,s)$; a →

Bob: let $M'=M$.
1.Computes $h=H(M||r)$.
>> h=concat(M,r)
2.Verifies signature on h.

Non-Interactive Zero Knowledge Proof - NIZKP.

Alice using the Schnorr Signature scheme can prove a knowledge of any other secret in one or other way related with Discrete Exponent Function - DEF.

Let this secret is some integer ***i*** and then **Alice** using DEF computes so called **Statement** we denote by ***t*** for her secret ***i***:

$$t = g^i \bmod p.$$

In this scenario Alice is called a ***Prover***.

Then **Alice** realizes a NIZKP of knowledge of ***i*** without revealing ***i*** by presenting this **Statement** ***t*** to the ***Verifier*** **Bob**.

$A: u \leftarrow randi(\mathbb{Z}_{p-1})$; $\mathbb{Z}_{p-1} = \{0, 1, 2, \dots, p-2\}$; $+, -, *, ^{}$ $mod(p-1)$

$r = g^u \bmod p$

$h = H(a||t||r)$

$s = u + ih \bmod (p-1)$

$\Pi = (r, s)$

$\Pi = (r, s), t, a$ →

$B: h = H(a||t||r)$

$g^s = r \cdot t^h \bmod p$

$g^s = g^{u+ih \bmod (p-1)} \bmod p = g^u \cdot g^{ih} \bmod p = r \cdot (g^i)^h = r \cdot t^h \bmod p$

Mini-https

Hand shaking PuK_A ⟷

AKAP ⟷

$G, \sigma = (r, s)$ ⟷

$PrK_B = y =$ randi(p-1)

$PuK_B = b = g^y \bmod p$

$PrK_A = x =$ randi(p-1)

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

$PuK_A = a$

$u \leftarrow randi(p-1)$

>> u = int64(randi(p-1))

$t_A = g^u \bmod p$

>> tA = mod_exp(g, u, p)

$Sign(x, t_A) = \sigma_A = (r_A, s_A)$

$t_A, \sigma_A = (r_A, s_A)$ → PuK_A

$i \leftarrow randi(p-1)$

$r_A = g^i \bmod p$

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

>> con = concat(t_A, r_A)

>> hA = hd28(con)

$s_A = (i + x \cdot h_A) \bmod (p-1) = \sigma_A = (r_A, s_A)$

t_B, σ_B ← PuK_B

1. Verify if PuK_A is in Data Base.

2. Verify if σ_A on t_A is valid.

Ver(PuK_A, σ_A, t_A) = T

3. Generates $v \leftarrow randi(p-1)$

computes $t_B = g^v \bmod p$.

$Sign(y, t_B) = \sigma_B = (r_B, s_B)$

(R, S)

Alice is verifying Bank's signature on t_B.
Ver(σ_B=(r_b, s_b), t_B) = **True**

$k_{AB} = (t_B)^u \bmod p =$
$= (g^v)^u \bmod p = g^{uv} \bmod p = k_{AB} = k = k_{BA}$

$k_{BA} = (t_A)^v \bmod p =$
$= (g^u)^v \bmod p = g^{uv} \bmod p$

A: creates transaction T_x

$E(k, T_x) = C$

$j \leftarrow randi$

r = g^j mod p

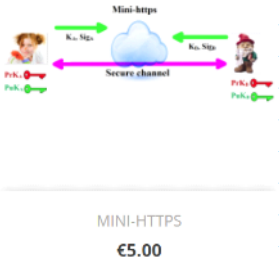
$h = H(C \| r)$

$s = j + x\,h \bmod (p-1)$

$\sigma = (r, s)$

$C, \sigma = (r, s)$ →

1. $Ver(PuK_A = a, \sigma, C) = \{T, F\}$
2. $D(k, C) = Tx$
3. Performes money transf.



MINI-HTTPS
€5.00

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

$$h = H(C \| r),$$

and verifies if

$$g^s \bmod p = r a^h \bmod p.$$

V1 V2

Symbolically this verification function we denote by

Ver(*a*,σ,*h*)=V∈{*True*, *False*}≡{1, 0}.

This function yields *True* if (2.22) is valid if:

PuK_A= *a* =*F*(PrK_A)= g^x *mod p*

1. Mentor sends you Public Parameters (***p***=268435019; ***g***=2) of 28 bits length.
Generate public and private keys **PrK_A=*x*** and **PuK_A=*a***.
Send public key [***a***] to the Mentor.

```
>> a
a = 39794323
```

39794323

2. Compute random secret number ***u*** of 28 bit length and compute session public parameter t_A. Sign t_A with Schnorr signature scheme by computing two signature components σ_A=(r_A, s_A). Send [tA,rA,sA] to the Mentor.

```
>> u=int64(randi(p))
u = 190442301
>> tA=mod_exp(g,u,p)
tA = 109819746
>>
>> i=int64(randi(p))
i = 236712983
>> rA=mod_exp(g,i,p)
rA = 117664796
>> con=concat(tA,rA)
con = 109819746117664796
```

109819746,117664796,114109665

```
% Alice verifies her signature before sending to Mentor
>> g_s=mod_exp(g,s,p)
g_s = 254713335
```

% Alice verifies her signature before sending to Mentor
>> g_s=mod_exp(g,s,p)
g_s = 254713335
>> a_h=mod_exp(a,h,p)
a_h = 85497572
>> V1=g_s
V1 = 254713335
>> V2=mod(r*a_h,p)
V2 = 254713335

rA = 117664796
>> con=concat(tA,rA)
con = 109819746117664796
>> h=hd28(con)
h = 243151357
>> sA=mod(i+x*h,p-1)
sA = 114109665

3. Mentor sends you (t_B **PuK$_B$**=32768, K$_B$ t_B=209399419, R=101644938, S=18011748). Verify Mentor's signature $\sigma_M=(R,S)$ on t_B. If signature is valid then taking S compute verification parameter $V_1=g^S \bmod p$. Compute common symmetric secret key **k** and transform **k** to the hexadecimal form ***kh*** of 32 digits length as it is required for AES128 function. Create the string of message variable ***m***='MMDD' consisting of the month and day of your birth. Encrypt message ***m*** using 1 round of AES128 cipher with key ***kh***32 by computing ciphertext >> C=AES128(m,kh32,1,'e'). **Attention**! Encryption using 1 round is extremely insecure and is used there to speed up the computations and to make sure of its insecurity. Insecurity is seen by comparing plaintext and ciphertext messages in hexadecimal format. They have non-excrypted digits. ***C*** should be entered within ' '. Send [V1,C] to the Mentor for decryption.

>> PuKB=int64(32768)
PuKB = 32768
>> tB=int64(209399419)
tB = 209399419
>> R=int64(101644938)
R = 101644938
>> S=int64(18011748)
S = 18011748
>>
>> con=concat(tB,R)
con = 209399419101644938
>> h=hd28(con)
h = 64128805

% 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
%
% fun - letter determining either encription: fun='e' or decryption: fun='d' functions

% Alice verifies her signature
>> g_S=mod_exp(g,S,p)
g_S = 20703551
>> V1=g_S
V1 = 20703551
V2 = 20703551

% Alice computes common
% symmetric secret key ***k***
>> k=mod_exp(tB,u,p)
k = 63198998

>> kh32=dec2hex(k,32)
kh32 = 0000000000000000000000003C45716
>> m='1012'
m = 1012
>> NR=1
NR = 1
>> C=AES128(m,kh32,NR,'e')
new = ~8$M~8t ~[illegible] =[illegible]
C = 7e38244d7e3874187ee424183dfc730e

20703551,'7e38244d7e3874187ee424183dfc730e'

4. Ok, let be informed that Mentor gets you a price for your birthday. The sum of the price he is sending to you as a ciphertex

>> CM='7e38245d7e38d8187e47241865fc730e'
CM = 7e38245d7e38d8187e47241865fc730e

4. Ok, let be informed that Mentor gets you a price for your birthday. The sum of the price he is sending to you as a ciphertex (***CM***='7e38245d7e38d8187e47241865fc730e'). Please decrypt and check it and then encrypt it again with added string 'ok' right after the sum by computing ciphertext ***C***1.
Send [C1] to the Mentor. ***C***1 should be entered within ' '.

'7e3824847e3840187efa24189efc730e'

Check Result

Success! You have finished the task. Great job!

Get reward

>> CM='7e38245d7e38d8187e47241865fc730e'
CM = 7e38245d7e38d8187e47241865fc730e
>>
>> M=AES128(CM,kh32,NR,'d')
Out = 00000000000000000000000000003935
M = 95
>>
>> Mok='95ok'
Mok = 95ok
>> C1=AES128(Mok,kh32,NR,'e')
new = ~8$8@ ~
C1 = 7e3824847e3840187efa24189efc730e

Till this place