https://imimsociety.net
https://imimschool.net

The results of mid-term-exam are entered AIS.
Course works list:
https://docs.google.com/document/d/11Bwk8HXLvjvzAEImcRiFcacwnrrz0lBs/edit?usp=sharing&ouid=111502255533491874828&rtpof=true&sd=true

Mini-https: https://imimsociety.net/en/14-cryptography
To provide information exchange **Confidentiality**, **Integrity** and **Authenticity** even using open communication channels.

MINI-HTTPS
€5.00

Public Parameters PP = (p, g).

p= 268435019; g=2;

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

A

Hand shaking PuK_A

AKAP, (E,D), (Sign,Ver)

AKAP

C, $\sigma = (r, s)$

B

PrK_B = y = randi(p-1)
PuK_B = B = g^y mod p

PrK_A = x = randi(p-1)
PuK_A = a = g^x mod p
k

PuK_A = a
k

$h_c = H(C \| r)$

$E(k, Tx) = C$

Encrypt & sign paradigm

Chosenc Ciphertex Attack
CCA

1. $Ver(PuK_A = a, \sigma, h_c) = \{T, F\}$

2. $D(k, C) = Tx$

3. Performes money transf.

By realizing Schnorr-Sign

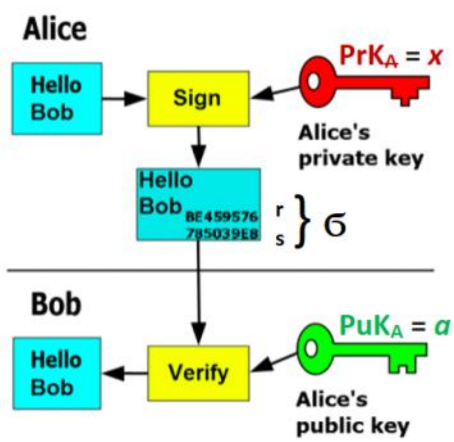
$i \leftarrow randi(p-1)$

$r = g^i \bmod p$

$h_c = H(C \| r)$

$Sign(PrK = x, h_c) = \sigma = (r, s)$

$s = (i + x \cdot h_c) \bmod (p-1)$



https://imimschool.net

https://imimschool.net/en/home/50-mini-https.html

Go back

Mini-https

All problems must be bought sequentially only one problem at time. It is strongly recommended that you use Octave software implemented in your local computer. The Octave installation version compatible with our .m files can be downloaded from: http://crypto.fmf.ktu.lt/xdownload/

This protocol is executed between Student Alice (you) and Mentor Bob. Both parties want to create secure channel where all messages exchanged by Alice and Bob will be encrypted using AES128() symmetric encryption function. To do that parties are realizing Authenticated Key Agreement Protocol (AKAP) based on Schnorr signature. Then the messages exchanged by Alice and Bob are encrypted using AES128() with agreed symmetric secret session key.

In this realization, encryption is performed for 1 block plain text to be short and to simplify computations. The length of this block consist of 128 bits which corresponds to 16 characters encoded in ASCII format (8 bits for 1 character). All plaintext messages to be encrypted are assigned to the string type variables, e.g. >> m=m='Hello Bob'; thus after encryption, obtained ciphertext CC is represented in two ways:

1) by ASCII encoded string type variable consisting of 16 characters (16x8=128 bits), and

2) by hexadecimally encoded string consisting of 32 digits (4 bits for 1 digit: 4x32=128 bits).

Parameter values sent by Alice must be included in input field separated by commas. In this instruction only input field is denoted by brackets [ ]. E.g., if Alice under the Mentor's request computed public parameters p=201623207 and g=22, then symbolically in the input field she must enter [p,g] which corresponds to enter 201623207, 22 in the input field sent by Mentor without any brackets.

Mentor's public key is ***B*****.**

The following functions are used in the protocol:

- >> $int64(randi(rng))$
  % Returns random integer i of 64 bits length in range$0 < i < rng$.
- >> $mod\_exp(g,x,p)$
  % Returns $a=g^x \ mod \ p$, check that $1 < x < p-1$, and $1 < g < p-1$.
- >> $hd28(strg)$
  % Returns 28-bits decimal h-value when input is symbolic variable $strg$.
  %Variable $strg$ can be fomed by $concat$ function;
- >> $C = AES128(m, Kh, NR,' e')$
  % $AES128$ encryption function returning ciphertext C encoded ASCII and hexadecimal formats;
  %Encryption is performed for 1 plaintext block of 128 bits length corresponding to 16 ASCII symbols;
  % m - message of string type to be encrypted, e.g. >> $m =' HelloBob'$;
  % K_h - secret symmetric key represented in hexadecimal digits % NR - number of rounds used in $AES128$;
  % $'e'$ - string type variable calling encryption function;
- >> $m = AES128(C, key, Nr,' d')$
  % $AES128$ decryption function returning decrypted message D of string type;
  % $'d'$ - string type variable calling decryption function;

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.

149403

```
>> p=int64(268435019)
p = 268435019
>> g=2
g = 2
>> x=int64(randi(p-1))
x = 96302305
>> a=mod_exp(g,x,p)
a = 149403
```

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 $\sigma=(r,s)$. Send $[t_A, r, s]$ to the Mentor.

8797451, 139674, 79705190

```
>> u=int64(randi(p-1))
u = 130744496
>> tA=mod_exp(g,u,p)
tA = 8797451
>> i=int64(randi(p-1))
i = 219891350
>> r=mod_exp(g,i,p)
r = 139674
>> cc=concat(tA,r)
cc = 8797451139674
>> h=hd28(cc)
h = 227536464
>> s=mod(i+x*h,p-1)
s = 79705190
```

3. Mentor sends you (t_B=32768, K_B =240219802, R =124553424, S =173651013). Verify Mentor's signature $\sigma_M=(R,S)$ on t_B. If signatur is valid then taking S compute verification parameter $V_1=g^S$ $mod\ p$. Compute common symmetric secret key k and transform k to the hexadecimal form k_h 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 k_h by computing ciphertext
>> C=AES128(m,Kh,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 $[V_1, C]$ to the Mentor for decryption.

208873585, '78bea39a78be59c878e5a3c8a7fd6ad7'

B t_B

```
>> B=int64(32768)
B = 32768
>> tB=int64(240219802)
tB = 240219802
>> R=int64(124553424)
R = 124553424
>> S=int64(173651013)
S = 173651013
>> g_S=mod_exp(g,S,p)
g_S = 208873585
V1=g_S;
>> Con=concat(tB,R)
Con = 240219802124553424
>> H=hd28(Con)
H = 250537502
>> B_H=mod_exp(B,H,p)
B_H = 109471541
>> RB_H=mod(R*B_H,p)
RB_H = 208873585
V2=RB_H;
```

Verification:

$$g^S \bmod p = RB^H \bmod p.$$
$$V1 = V2$$

```
>> m='0501'
m = 0501
>> k=mod_exp(tB,u,p)
k = 239323423
>> kh=dec2hex(k,32)
kh = 000000000000000000000000E43C91F
>> C=AES128(m,kh,1,'e')
new = xxYxàj
C = 78bea39a78be59c878e5a3c8a7fd6ad7
```

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 (C_M ='78bea3da78be81c878c7a3c8befd6ad7'). Please decrypt and check it and then encrypt it again with added string 'ok' right after the sum by computing ciphertext C_1.
Send $[C_1]$ to the Mentor. C_1 should be entered within ''.

'78bea33978bee4c8788ca3c860fd6ad7'

Check Result

Success! You have finished the task. Great job!

Get reward

```
>> CM='78bea3da78be81c878c7a3c8befd6ad7'
CM = 78bea3da78be81c878c7a3c8befd6ad7
>> M=AES128(CM,kh,1,'d')
Out = 00000000000000000000000000003133
```
M = 13
```
>> Mok='13ok'
Mok = 13ok
>> CMok=AES128(Mok,kh,1,'e')
new = x9xx`j
CMok = 78bea33978bee4c8788ca3c860fd6ad7
```

Let the number of rounds is equal to NR = 15.

```
>> m = '0501'
m = 0501
>> M_ok='64ok'
M_ok = 64ok

>> NR=10
NR = 10
>> C15=AES128(m,kh,NR,'e')
new = \DX-e    C
C15 = 5c448d582d6520850001fb439eaa9bc6
>> C15C1=AES128(M_ok,kh,NR,'e')
new = ӜvY(ن
C15C1 = a785bb8bb3d9d39c765928bdd986fdef
```