El Gamal E-Signature
€3.00

ElGamal Encryption
€3.00

Schnorr E-Signature
€3.00

Schnorr Identification
€3.00

MINI-HTTPS
€5.00

Mini-https

Confidential, Integral, Authentic

Public Parameters PP = (p, g).

p= 268435019; g=2;

(E, D), $(\mathrm{Sign}, \mathrm{Ver})$

A

Hand shaking $\xleftrightarrow{PuK_A}$

$\xleftrightarrow{AKAP}$

$\xleftrightarrow{G,\ \sigma=(r,s)}$

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

$u \leftarrow \mathrm{randi}(p-1)$

>> $u = \mathrm{int64}(\mathrm{randi}(p-1))$

$t_A = g^u \bmod p$

>> $t_A = \mathrm{mod\_exp}(g, u, p)$

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

$i \leftarrow \mathrm{randi}(p-1)$

$r_A = g^i \bmod p$

$t_A,\ \sigma_A = (r_A, s_A) \xrightarrow[PuK_A]{}$

1. Verify if PuK_A is in Data Base.
2. Verify if σ_A on t_A is valid.
$\mathrm{Ver}(PuK_A, \sigma_A, k_A) = T$
3. Generates $v \leftarrow \mathrm{randi}(p-1)$
Computes $t_B = g^v \bmod p$.
$\mathrm{Sign}(y, t_B) = \sigma_B = (r_B, s_B)$

$\xleftarrow[PuK_B]{t_B,\ \sigma_B}$

>> con = concat(t_A, r_A)

>> h_A = hd28(con)

$s_A = (i + x \cdot h_A) = \sigma_A = (r_A, s_A)$

$k_{AB} = (t_B)^u \bmod p =$

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

$k_{BA} = (k_A)^v \bmod p =$

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

A: creates transaction T_x

$E(k, T_x) = C$

$j \leftarrow \text{randi}$

$r \leftarrow \text{randi}$

$h = H(C \| r)$

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

$\sigma = (r, s)$

$\xrightarrow{C,\ \sigma=(r,s)}$

1. $Ver(PuK_A = a, \sigma) = \{T, F\}$
2. $D(k, C) = T_x$
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

$$\underbrace{g^s \bmod p}_{V1} = \underbrace{r a^h \bmod p}_{V2}.$$

Symbolically this verification function we denote by

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

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

$$PuK_A = a = F(PrK_A) = g^x \bmod p$$

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**=(***r***, ***s***). Send [tA,r,s] to the Mentor.

```
>> u=int64(randi(p))
u = 190442301
>> tA=mod_exp(g,u,p)
tA = 109819746
>>
>> i=int64(randi(p))
i = 236712983
>> r=mod_exp(g,i,p)
r = 117664796
>> con=concat(tA,r)
con = 109819746117664796
>> h=hd28(con)
h = 243151357
>> s=mod(i+x*h,p-1)
s = 114109665
```

109819746,117664796,114109665

```
% 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
```

3. Mentor sends you (t_B **PuK**$_B$=32768, K_B t_B=209399419, ***R***=101644938, ***S***=18011748). Verify Mentor's signature σ_M=(***R***,***S***) on t_B. If signature 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 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

```
>> PuKB=int64(32768)
PuKB = 32768
>> tB=int64(209399419)
tB = 209399419
>> R=int64(101644938)
R = 101644938
>> S=int64(18011748)
S = 18011748
```

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.

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

% 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
```

```
% 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
```

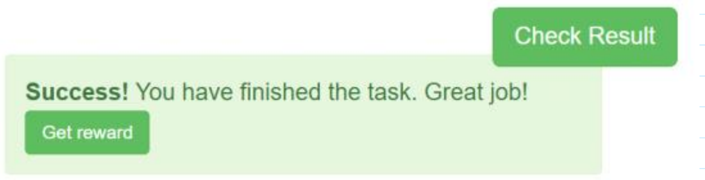
```
>>  kh32=dec2hex(k,32)
kh32 = 0000000000000000000000003C45716
>> m='1012'
m = 1012
>> NR=1
NR = 1
>> C=AES128(m,kh32,NR,'e')
new = ~8$M~8t  ~�  =�
C = 7e38244d7e3874187ee424183dfc730e
```

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'). 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 ' '.

```
>> 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
```

'7e3824847e3840187efa24189efc730e'



Success! You have finished the task. Great job!

Till this place